МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ
УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

Кафедра «Програмна інженерія»

ЗВІТ

до практичного завдання №1

з дисципліни «Аналіз та рефакторинг коду»

на тему «Правила оформлення програмного коду Kotlin»

Виконала:
ст. гр. ПЗПІ-23-7
Никифорова Анастасія

Перевірив:
ст. викладач катедри ПІ
Сокорчук Ігор Петрович

Харків 2025

ІСТОРІЯ ЗМІН

№	Дата звернення	Версія звіту	Опис змін та виправлень
1	15.01.26	1.1	Створено розділ «Завдання».
2	15.01.26	1.1	Додано розділ «Опис виконаної роботи».
3	15.01.26	1.1	Створено розділ «Висновки».
4	17.01.26	1.1	Додано розділ «Використані джерела».

2 ЗАВДАННЯ

Мета роботи

Ознайомитися з основними рекомендаціями щодо написання чистого, ефективного та підтримуваного коду мовою Kotlin, а також сформувати навички аналізу якості програмного коду.

Завдання

Дослідити основні рекомендації для написання якісного програмного коду мовою Kotlin, продемонструвати ці рекомендації на прикладах коду, обґрунтувати кожну рекомендацію та надати детальні пояснення. Для кожного прикладу повинні бути наведені відповідні фрагменти програмного коду. У якості обраної мови програмування використано Kotlin.

3 ОПИС ВИКОНАНОЇ РОБОТИ

Важливість правил оформлення коду (Kotlin)

Дотримання правил оформлення програмного коду є невід’ємною частиною професійної розробки програмного забезпечення мовою Kotlin. Чітко структурований, зрозумілий та уніфікований код значно спрощує його підтримку, розширення та тестування. Оскільки Kotlin широко використовується для розробки Android-застосунків та серверних рішень, дотримання єдиних стандартів стилю дозволяє ефективно працювати в команді та швидко орієнтуватися в проєкті.

Правильно оформлений код зменшує ймовірність виникнення помилок, полегшує процес налагодження та покращує читабельність логіки програми. Використання рекомендацій, запропонованих спільнотою Kotlin та компанією JetBrains, сприяє створенню якісного, надійного та масштабованого програмного забезпечення.

Структура коду в Kotlin

У мові програмування Kotlin рекомендовано чітко організовувати структуру проєкту, розділяючи код на логічні пакети та файли. Кожен клас або файл має виконувати одну конкретну задачу, що відповідає принципу єдиної відповідальності. Назви пакетів зазвичай відповідають ієрархії проєкту та записуються у нижньому регістрі.

Файли Kotlin можуть містити як класи, так і функції верхнього рівня, що дозволяє уникати надмірного використання допоміжних класів. Логічно пов’язані компоненти групуються в одному пакеті, а великі проєкти поділяються на модулі. Такий підхід покращує навігацію в коді, спрощує супровід проєкту та забезпечує зручну масштабованість.

Форматування коду

Форматування коду в Kotlin відіграє важливу роль у забезпеченні його читабельності та підтримуваності. Рекомендовано використовувати відступи розміром у чотири пробіли, а не табуляцію, для уникнення різного відображення коду в різних середовищах розробки. Довжина рядка не повинна перевищувати 120 символів, що дозволяє зручно читати код без горизонтальної прокрутки.

Фігурні дужки в Kotlin розташовуються на одному рядку з оголошенням класу, функції або умовного оператора. Між логічними блоками коду рекомендується залишати порожні рядки для кращого візуального сприйняття. Також важливо використовувати пробіли навколо операторів та після ком, що робить код більш охайним і зрозумілим.

```
fun sum(a:Int,b:Int):Int{
return a+b
}
```

Рисунок 1.1 – Приклад неправильного форматування коду

```
fun sum(a: Int, b: Int): Int {
    return a + b
}
```

Рисунок 1.2 – Приклад правильного форматування коду

Іменування (змінні, функції, класи) в Kotlin

Іменування в мові програмування Kotlin є одним із ключових аспектів написання чистого та зрозумілого коду. Правильно підібрані імена змінних, функцій і класів дозволяють легко зрозуміти призначення елементів програми без детального аналізу їх реалізації.

Для іменування змінних та функцій у Kotlin використовується стиль camelCase, у якому перше слово починається з малої літери, а кожне наступне — з великої. Назви класів, інтерфейсів та об’єктів оформлюються у стилі PascalCase, де кожне слово починається з великої літери. Константи, як правило, записуються великими літерами з використанням символу підкреслення.

Дотримання стандартів іменування сприяє єдності стилю коду, зменшує ймовірність помилок та полегшує процес супроводу і масштабування програмного забезпечення.

```
class user {
    fun calc(x: Int): Int {
        return x * 2
    }
}
```

Рисунок 1.3 – Приклад неправильного іменування

```
class UserCalculator {

    fun calculateValue(inputValue: Int): Int {
        return inputValue * 2
    }
}
```

Рисунок 1.4 – Приклад правильного іменування

Коментарі та документування коду

Коментарі в програмному коді використовуються для пояснення логіки роботи програми та причин прийняття певних рішень. Вони мають відповідати на питання “чому”, а не “що”, оскільки добре написаний код зазвичай є самодокументованим. Надмірне коментування очевидних дій може ускладнювати читання коду.

У Kotlin використовуються однорядкові коментарі // для коротких пояснень та багаторядкові коментарі /* ... */ для опису великих логічних блоків. Особливу роль відіграє документація KDoc, яка починається з /** ... */ і дозволяє формально описувати функції, параметри та значення, що повертаються. Така документація інтегрується з середовищем розробки та значно полегшує супровід проєкту.

```kotlin
// Обчислення суми двох чисел
fun sum(a: Int, b: Int): Int {
    return a + b
}

/*
 Функція множення двох чисел.
 Використовується для базових математичних операцій.
 */
fun multiply(a: Int, b: Int): Int {
    return a * b
}
```

Рисунок 1.5 – Приклади коментування та документування

```kotlin
/**
 * Обчислює загальну вартість замовлення.
 *
 * @param price ціна одного товару
 * @param quantity кількість товарів
 * @return загальна вартість замовлення
 */
fun calculateTotal(price: Double, quantity: Int): Double {
    return price * quantity
```

Рисунок 1.6 – Приклад KDoc

Тестування та забезпечення якості коду

Тестування є важливою складовою процесу розробки програмного забезпечення мовою Kotlin. Воно дозволяє перевірити коректність роботи окремих компонентів програми, своєчасно виявити помилки та зменшити

ризик їх появи на пізніх етапах розробки. Найбільш поширеним видом тестування є модульне тестування (unit testing), яке зосереджується на перевірці окремих функцій або класів.

У Kotlin для модульного тестування часто використовуються бібліотеки JUnit та Kotlin Test. Тести дозволяють автоматично перевіряти очікувані результати виконання коду та забезпечують стабільність програми під час внесення змін. Використання тестування сприяє підвищенню якості коду, покращенню його надійності та підтримуваності.

Рисунок 1.7 – Приклад модульного тесту

Інструменти для підтримки якості коду

Для підтримки високої якості програмного коду мовою Kotlin використовуються спеціальні інструменти, які допомагають автоматично перевіряти стиль, форматування та можливі помилки. Одним з основних середовищ розробки для Kotlin є IntelliJ IDEA, яке надає вбудовані засоби аналізу коду та рекомендації щодо його покращення.

Для перевірки стилю та форматування коду застосовуються такі інструменти, як ktlint та detekt. Вони дозволяють виявляти порушення стандартів оформлення, потенційні помилки та складні для підтримки фрагменти коду. Використання подібних інструментів сприяє дотриманню єдиного стилю розробки та підвищує загальну якість програмного забезпечення.

4 ВИСНОВКИ

У ході виконання практичного завдання було розглянуто основні правила оформлення програмного коду мовою Kotlin. Проаналізовано важливість дотримання стандартів кодування, структуру проєкту, правила форматування, іменування змінних, функцій і класів, а також принципи коментування та документування з використанням KDoc.

У роботі також було розглянуто роль тестування у забезпеченні якості програмного коду та використання інструментів для автоматичного аналізу стилю й виявлення помилок. Наведені приклади коду дозволили наочно продемонструвати відмінності між правильним і неправильним оформленням.

У результаті виконання роботи сформовано розуміння важливості стандартів оформлення коду для підвищення його читабельності, підтримуваності та надійності. Отримані знання можуть бути використані під час розробки реальних програмних проєктів мовою Kotlin.

5 ВИКОРИСТАНІ ДЖЕРЕЛА

JetBrains. Kotlin Coding Conventions [Електронний ресурс]. – Режим доступу: https://kotlinlang.org/docs/coding-conventions.html (дата звернення: 2026).

JetBrains. Kotlin Documentation [Електронний ресурс]. – Режим доступу: https://kotlinlang.org/docs/home.html (дата звернення: 2026).

JUnit. Official Documentation [Електронний ресурс]. – Режим доступу: https://junit.org/junit5/ (дата звернення: 2026).

IntelliJ IDEA Documentation [Електронний ресурс]. – Режим доступу: https://www.jetbrains.com/idea/documentation/ (дата звернення: 2026).

ДОДАТОК А

https://youtu.be/zAYSLKmfC_E

ДОДАТОК Б

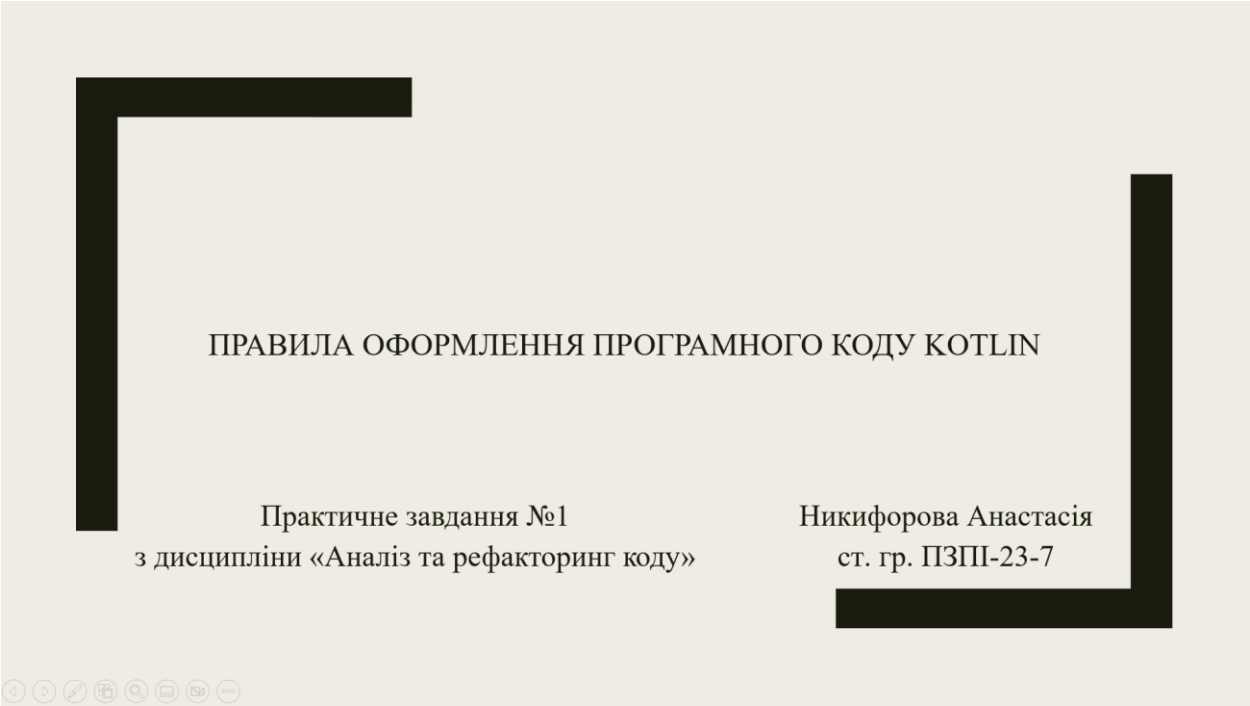

Рисунок Б.1 – Слайд 1 Вступний слайд

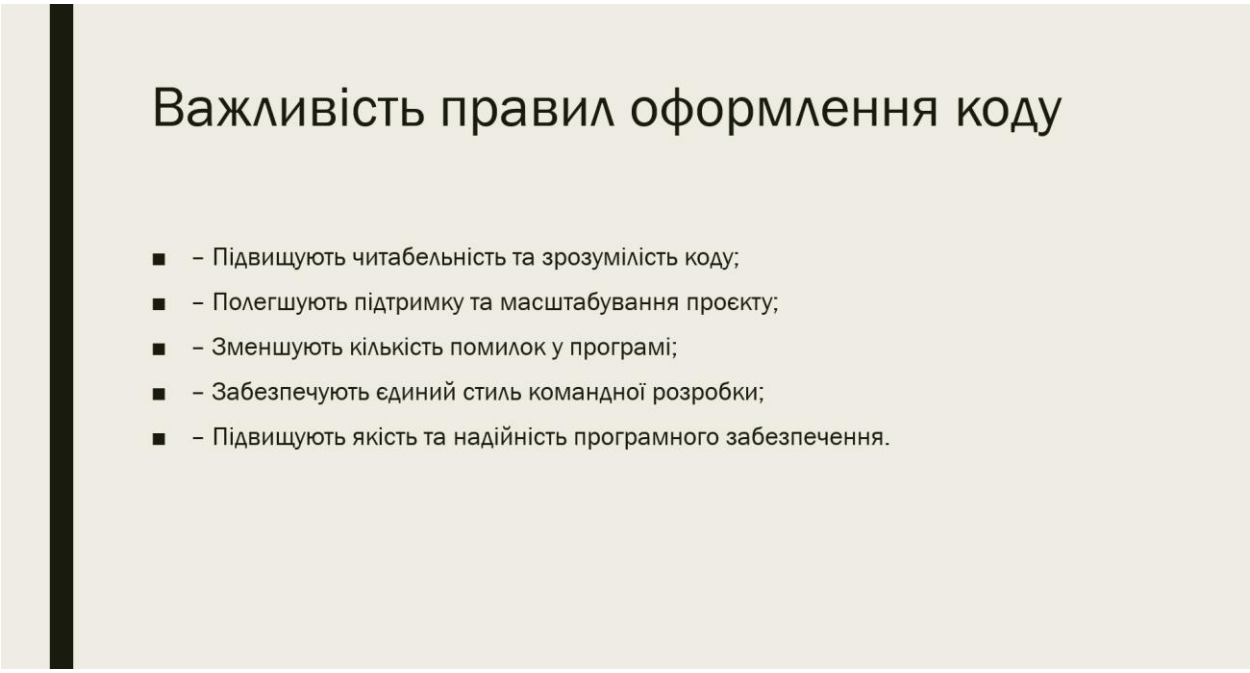

Рисунок Б.2 – Слайд 2 Опис важливості правил коду

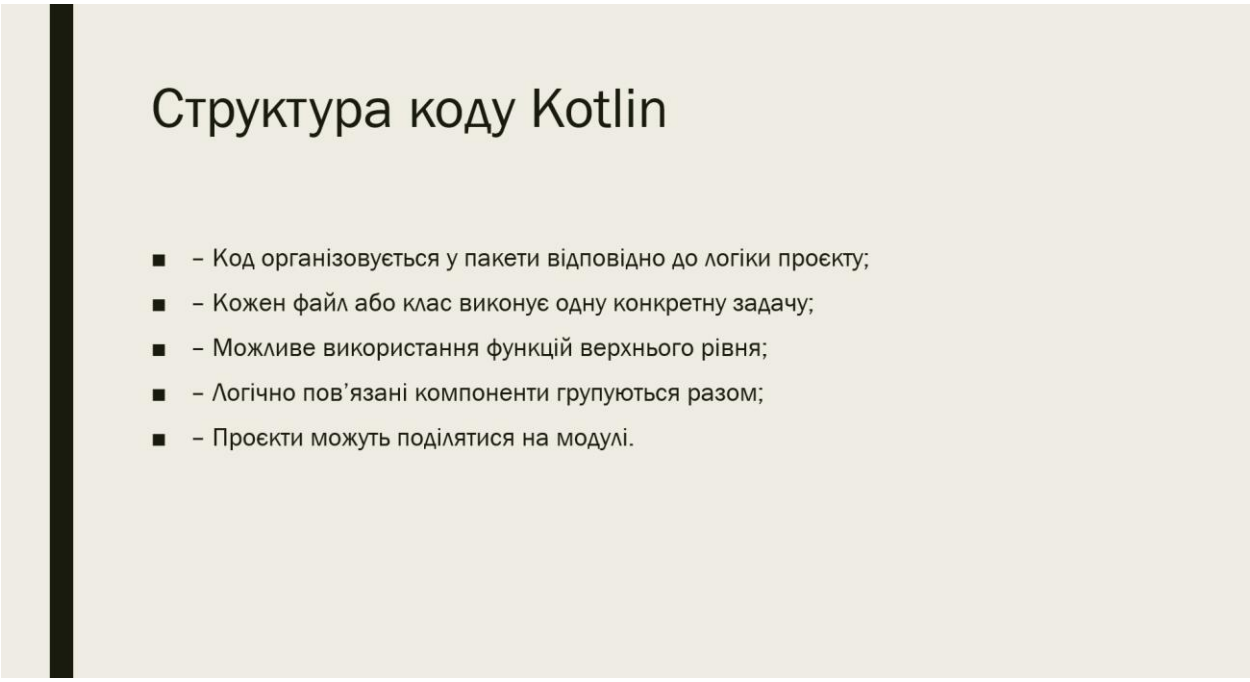

Рисунок Б.3 – Слайд 3 Опис структури коду

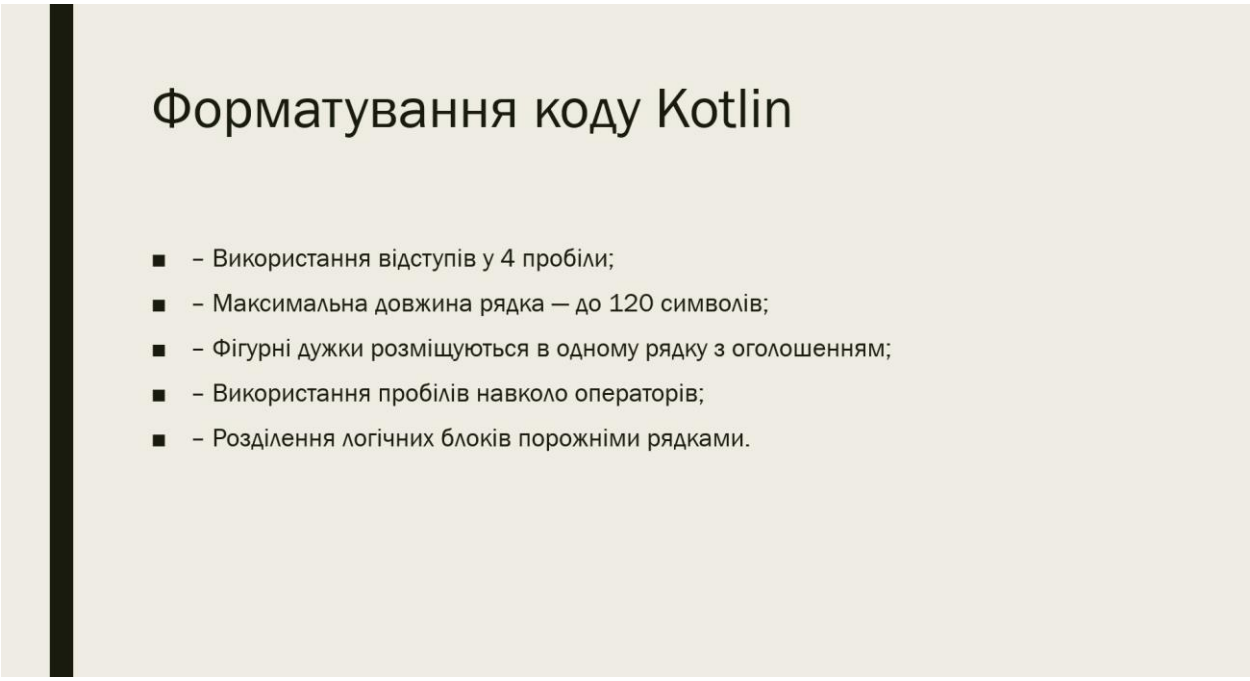

Рисунок Б.4 – Слайд 4 Опис правил форматування коду

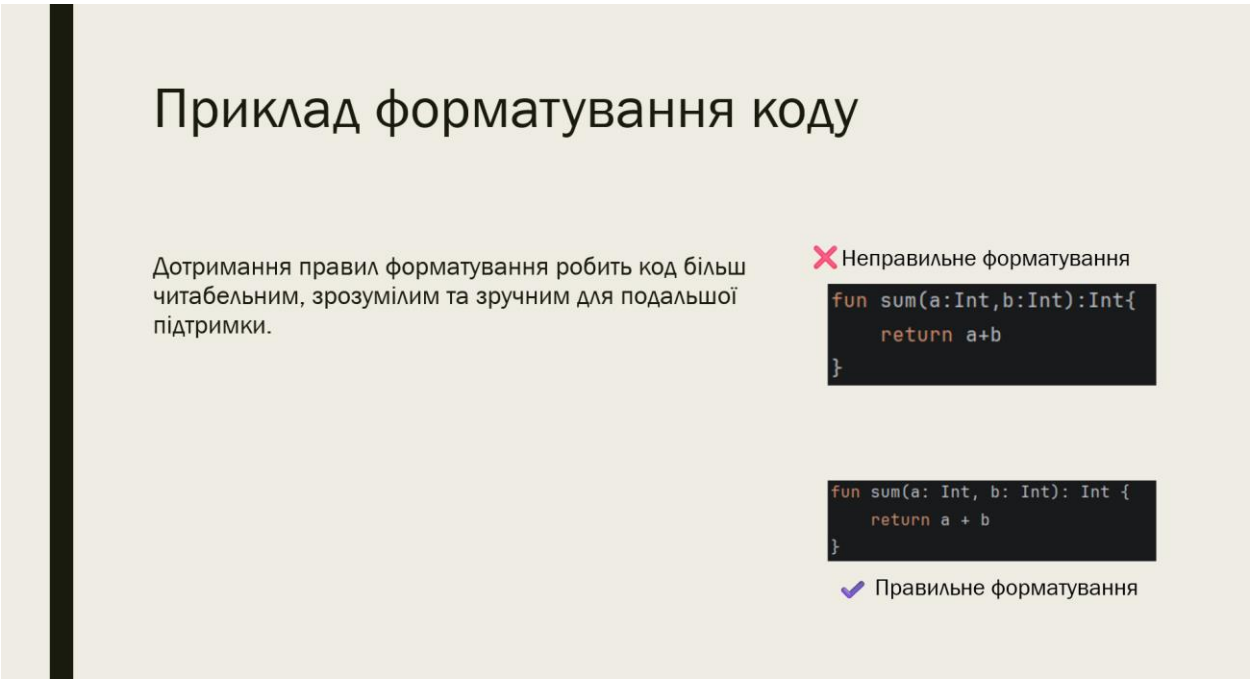

Рисунок Б.5 – Слайд 5 Приклад форматування коду

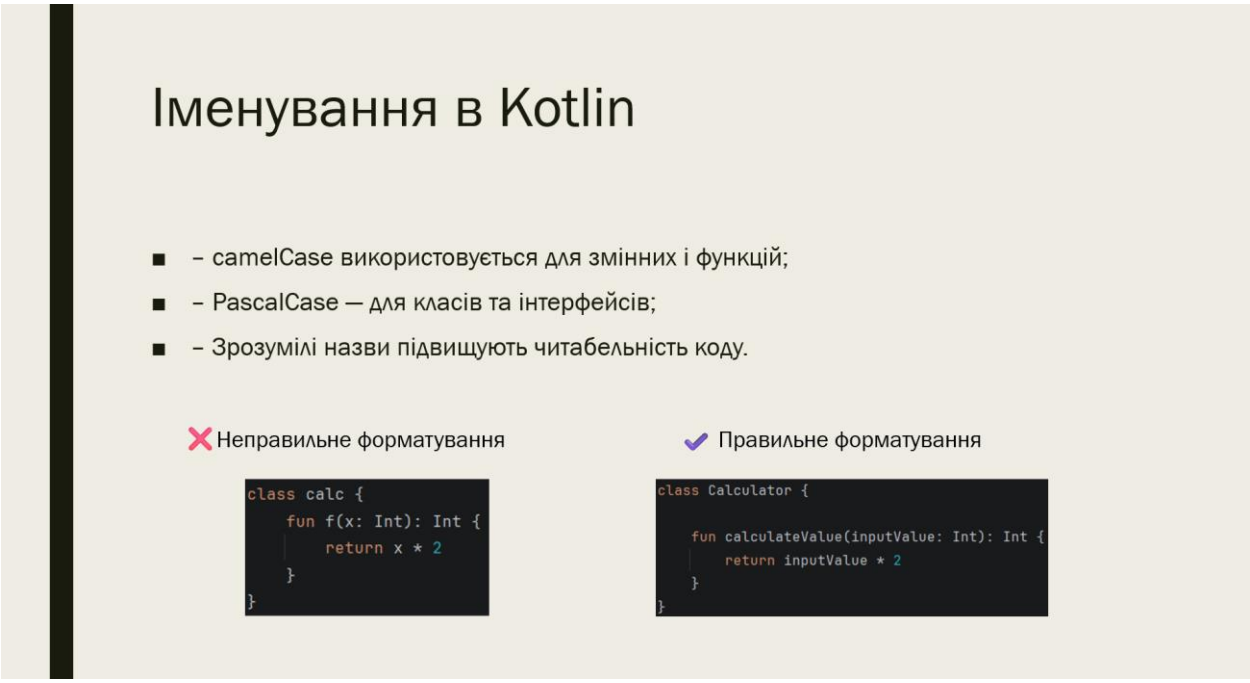

Рисунок Б.6 – Слайд 6 Опис правильного іменування та приклади

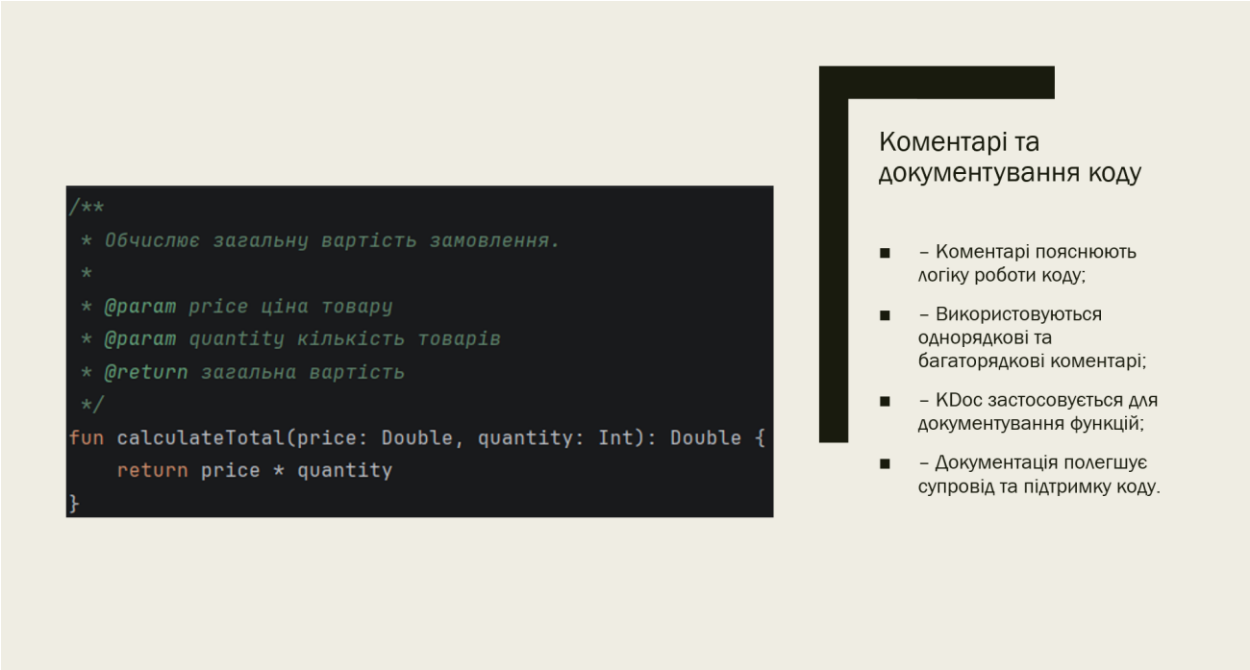

Рисунок Б.7 – Слайд 7 Опис правил коментарів, документування та приклад

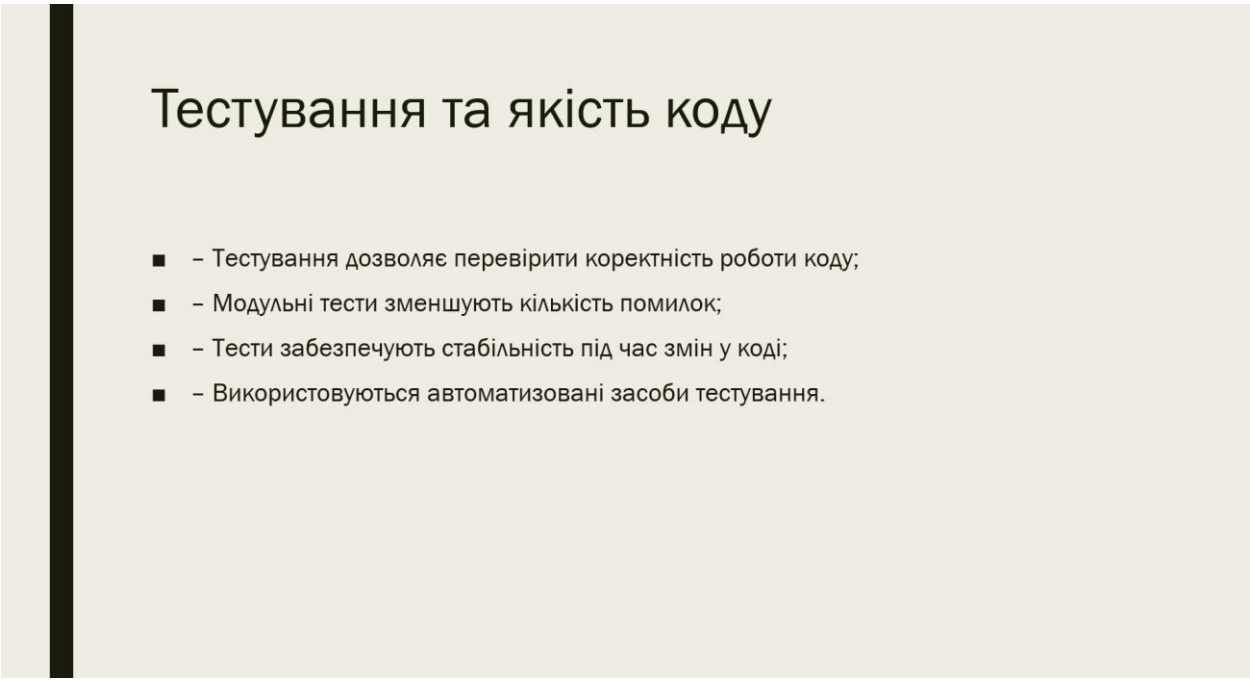

Рисунок Б.8 – Слайд 8 Опис тестування

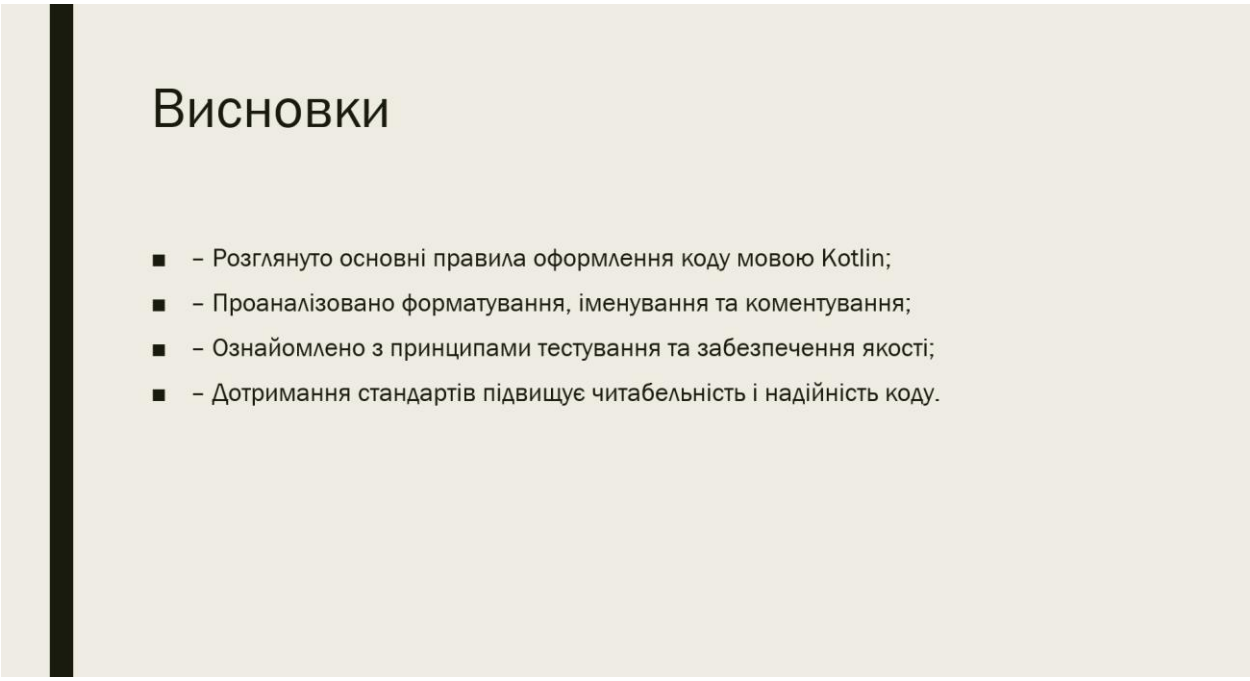

Рисунок Б.9 – Слайд 9 Висновки