

**МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ**  
**ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ**  
**РАДІОЕЛЕКТРОНІКИ**

**Звіт**  
**з практичної роботи №3**  
**із дисципліни «Аналіз та рефакторинг коду**  
**програмного забезпечення»**  
**на тему “ Refactoring Methods ”**

Виконав:

ПЗПІ-22-1 Тимощук Денис

Перевірив:

доцент кафедри програмної інженерії, к.т.н.

Лещинський Володимир Олександрови.

Харків 2024

## **Мета роботи**

Метою даної практичної роботи є засвоєння основних методів рефакторингу коду на основі реальних прикладів з моїх програмних проєктів. Я прагну навчитися ідентифікувати проблеми в коді та використовувати відповідні методи рефакторингу, такі як виділення методу, спрощення умовних виразів і заміна тимчасових змінних на запити, для покращення якості коду. Ця робота допомогла мені розвинути навички написання чистого, ефективного та підтримуваного коду, що є важливим аспектом програмування.

## **Висновки**

Застосовані методи рефакторингу значно покращили якість мого коду, зробивши його більш читабельним і легким для підтримки. Виділення методу (Extract Method) дозволило розділити складний код на окремі частини, що поліпшило його модульність і спростило тестування. Заміна тимчасових змінних на запити (Replace Temp with Query) усунула непотрібні змінні, що зробило код простішим і менш схильним до помилок. Спрощення умовних виразів (Simplify Conditional Expression) допомогло скоротити складну логіку, полегшуючи розуміння коду та підвищуючи його підтримку і масштабованість у майбутньому. Ці рефакторингові методи підкреслюють важливість підтримки чистоти та зрозумілості коду, що сприяє його ефективності та зручності в подальшій роботі.

# Додаток А

Скріншоти презентації представленої на практичному занятті

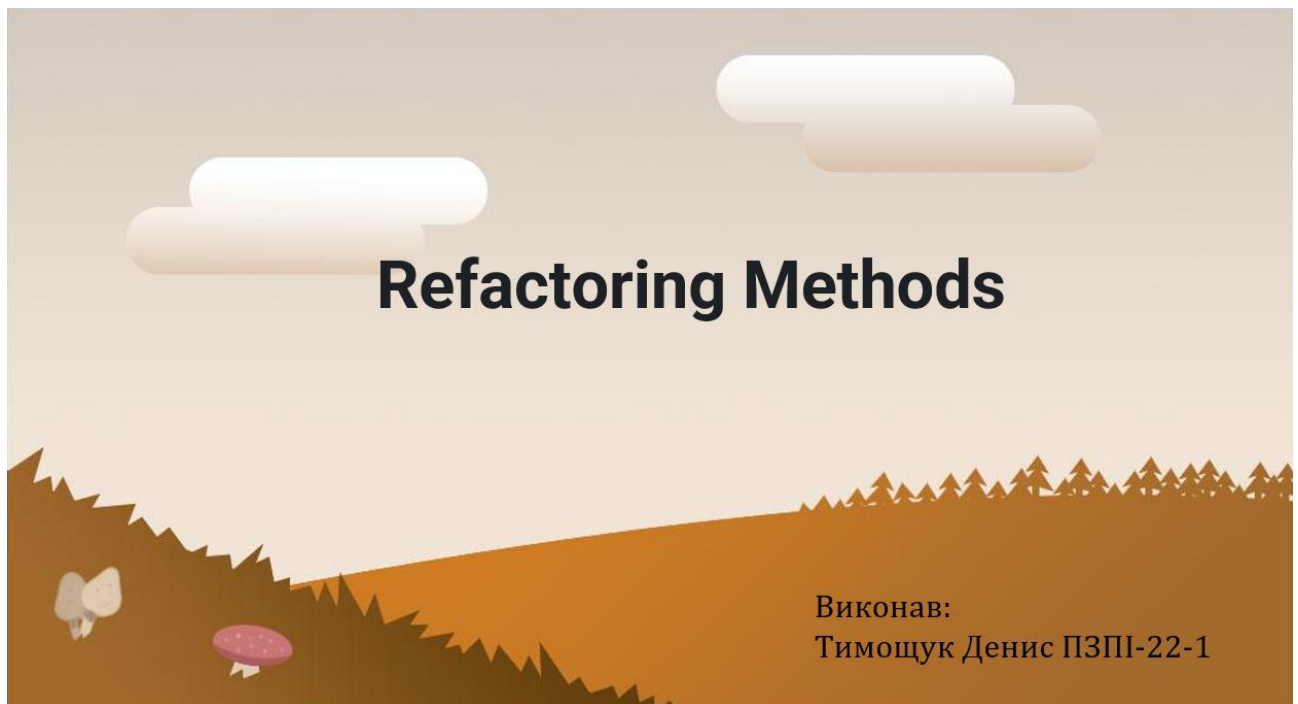
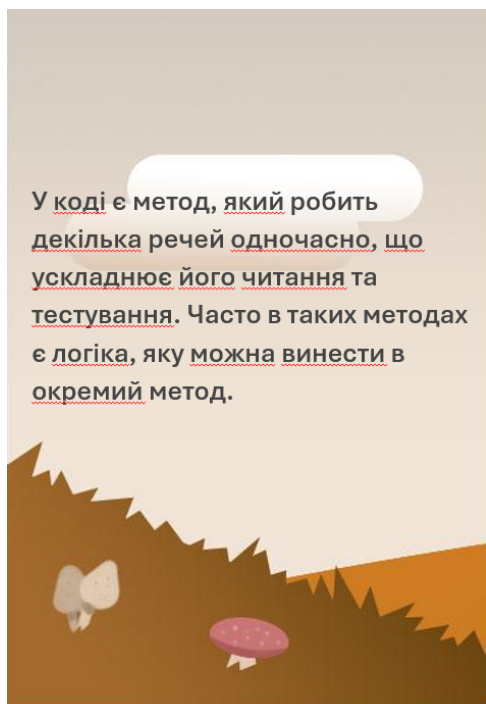


Рисунок 1 – Титульна сторінка



Extract Method (Виділення методу) До:

```
5 public void ProcessOrder(Order order)
6 {
7     // Перевірка наявності товарів на складі
8     foreach (var item in order.Items)
9     {
10         if (!IsInStock(item))
11         {
12             Console.WriteLine($"Item {item.Name} is out of stock.");
13         }
14     }
15
16     // Обрахунок загальної суми замовлення
17     decimal total = 0;
18     foreach (var item in order.Items)
19     {
20         total += item.Price;
21     }
22
23     // Виведення результату
24     Console.WriteLine($"Order total: {total}");
25 }
```

Рисунок 2 – Перший метод “Extract Method” до редагування

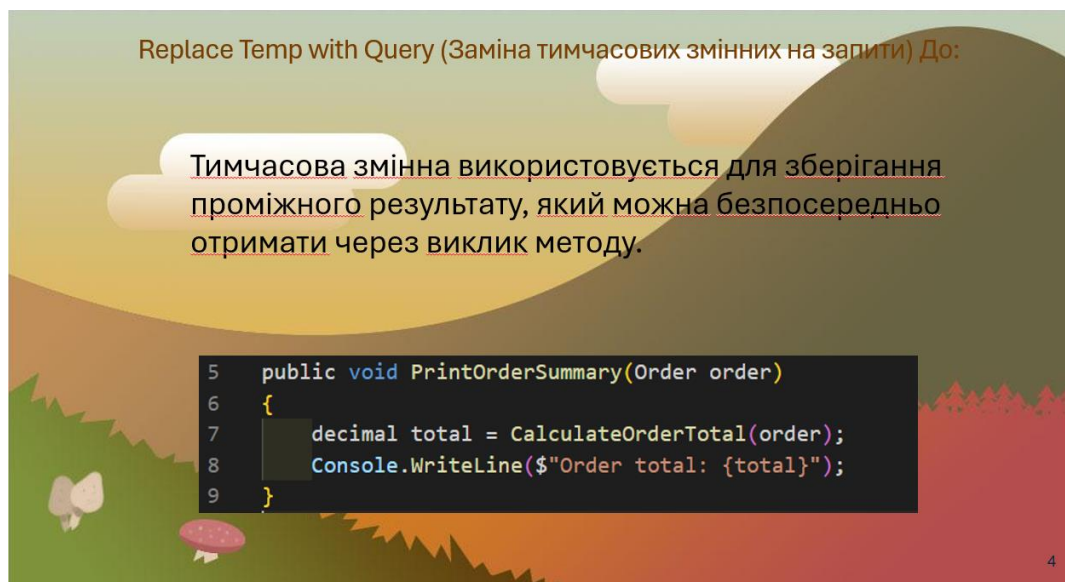


### Extract Method (Виділення методу) Після:

```
5 public void ProcessOrder(Order order)
6 {
7     CheckStock(order);
8     decimal total = CalculateTotal(order);
9     Console.WriteLine($"Order total: {total}");
10 }
11
12 private void CheckStock(Order order)
13 {
14     foreach (var item in order.Items)
15     {
16         if (!IsInStock(item))
17         {
18             Console.WriteLine($"Item {item.Name} is out of stock.");
19         }
20     }
21 }
22
23 private decimal CalculateTotal(Order order)
24 {
25     decimal total = 0;
26     foreach (var item in order.Items)
27     {
28         total += item.Price;
29     }
30     return total;
31 }
```

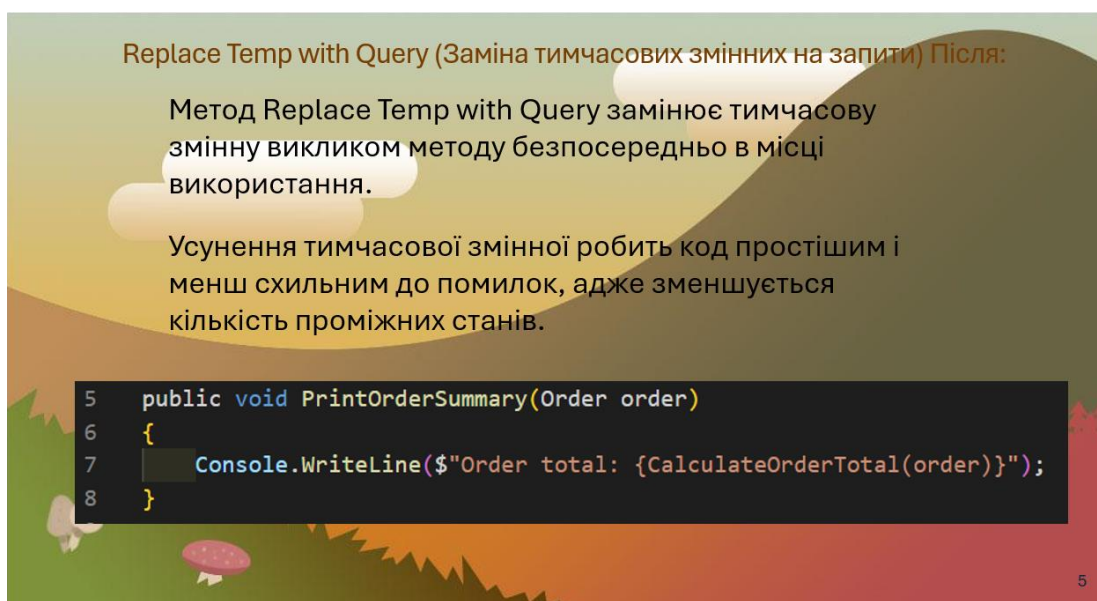
3

Рисунок 3 – Перший метод “Extract Method” після редагування



4

Рисунок 4 – Другий метод “Replace Temp with Query” до редагування



5

Рисунок 5 – Другий метод “Replace Temp with Query” після редагування

## Simplify Conditional Expression (Спрощення умовних виразів) До:

У коді є складний умовний вираз із великою кількістю умов або вкладених умов. Це ускладнює читання і підтримку коду.

```
5 public decimal GetDiscount(Customer customer)
6 {
7     if (customer.IsPremium)
8     {
9         if (customer.Age > 60)
10        {
11            return 0.2m; // 20% знижка для преміум-клієнтів віком понад 60 років
12        }
13        else
14        {
15            return 0.1m; // 10% знижка для преміум-клієнтів
16        }
17    }
18    else
19    {
20        if (customer.Age > 60)
21        {
22            return 0.05m; // 5% знижка для звичайних клієнтів віком понад 60 років
23        }
24        else
25        {
26            return 0.0m; // Немає знижки
27        }
28    }
29 }
```

6

Рисунок 6 – Третій метод “Simplify Conditional Expression” до редагування

## Simplify Conditional Expression (Спрощення умовних виразів) Після:

Метод Simplify Conditional Expression використовується для спрощення складних умовних блоків, зокрема, через виділення повторюваних частин або використання тернарних операторів.

Код став значно коротшим та простішим для розуміння. Умовний вираз більше не повторюється, а логіка знижки легше читається.

```
5 public decimal GetDiscount(Customer customer)
6 {
7     decimal baseDiscount = customer.Age > 60 ? 0.05m : 0.0m;
8     return customer.IsPremium ? baseDiscount + 0.1m : baseDiscount;
9 }
```

7

Рисунок 7 – Третій метод “Simplify Conditional Expression” після редагування

## Висновки

Застосовані методи рефакторингу значно покращили якість коду, зробивши його більш читабельним і легким для підтримки. Виділення методу (Extract Method) розділило складний код на окремі частини, що поліпшило його модульність і спростило тестування. Заміна тимчасових змінних на запити (Replace Temp with Query) усунула непотрібні змінні, зробивши код простішим і менш схильним до помилок, завдяки усуненню проміжних станів. Спрощення умовних виразів (Simplify Conditional Expression) скоротило складну логіку, полегшивши розуміння коду та підвищивши його підтримку і масштабованість у майбутньому.

8

Рисунок 8 – Слайд із висновками

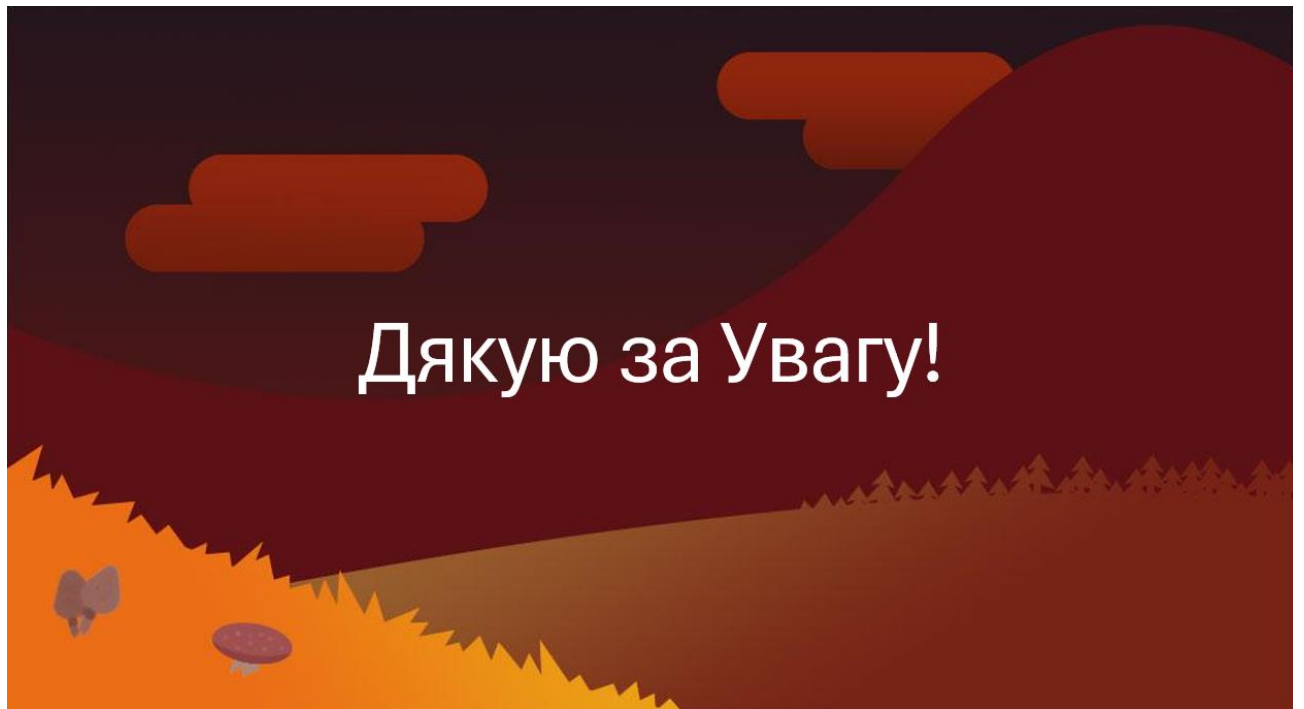


Рисунок 9— Слайд із подякою за увагу