Харківський університет радіоелектроніки
Факультет комп'ютерних наук
Кафедра програмної інженерії

ЗВІТ

до практичної роботи з дисципліни «Аналіз та рефакторинг коду»

на тему: «Методи рефакторингу коду програмного забезпечення»

Виконав ст. гр ПЗПІ-22-2 Верясов Владислав Олексійович

Перевірив
Доцент кафедри ПІ
Лещинський Володимир Олександрович

Харків 2024

МЕТА

Мета даної роботи полягала в дослідженні та застосуванні методів рефакторингу коду для покращення його структури, зменшення дублювання та підвищення читабельності. Особливий акцент було зроблено на методах "Pull Up Field", "Pull Up Method" та "Hide Delegate", які допомагають оптимізувати об'єктно-орієнтовані системи шляхом впорядкування спільних елементів і приховування внутрішніх деталей реалізації. Основною метою було вивчення способів покращення підтримки та розширюваності програмного забезпечення, збереження його якості та зменшення технічного боргу.

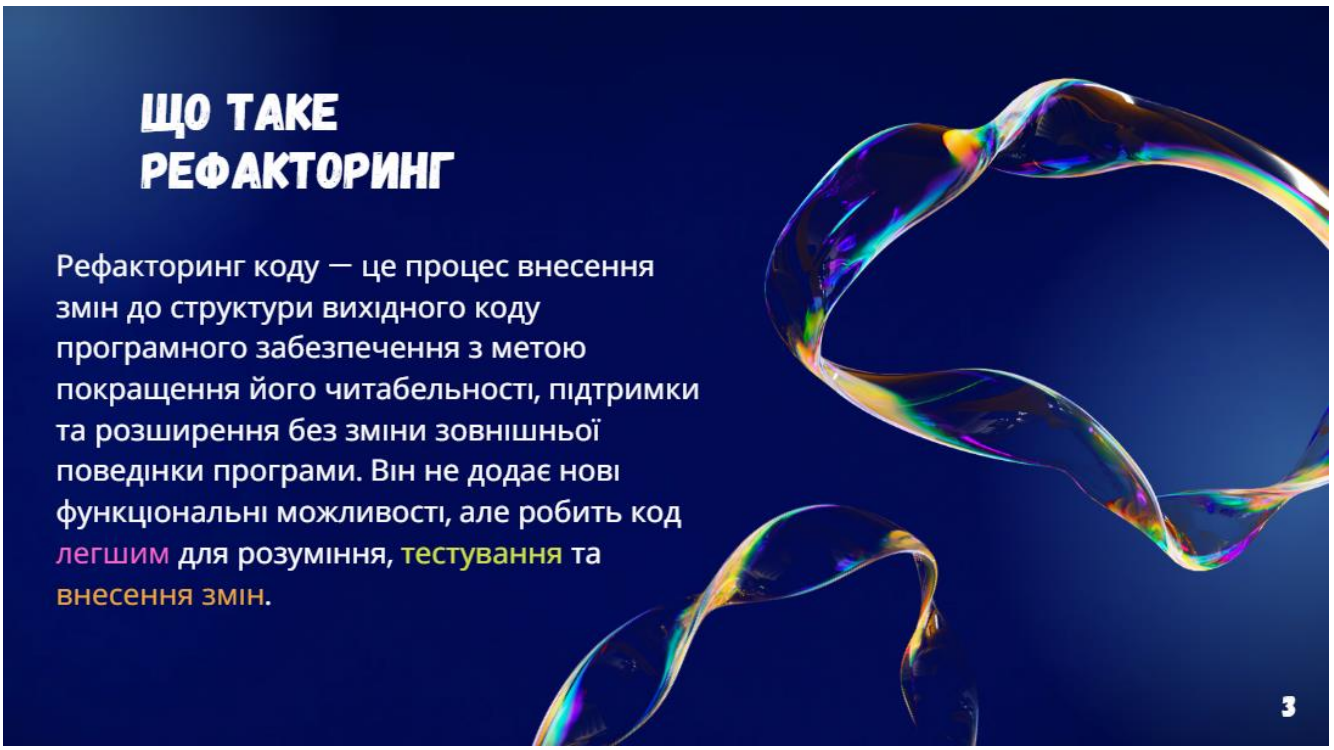
ЗАВДАННЯ

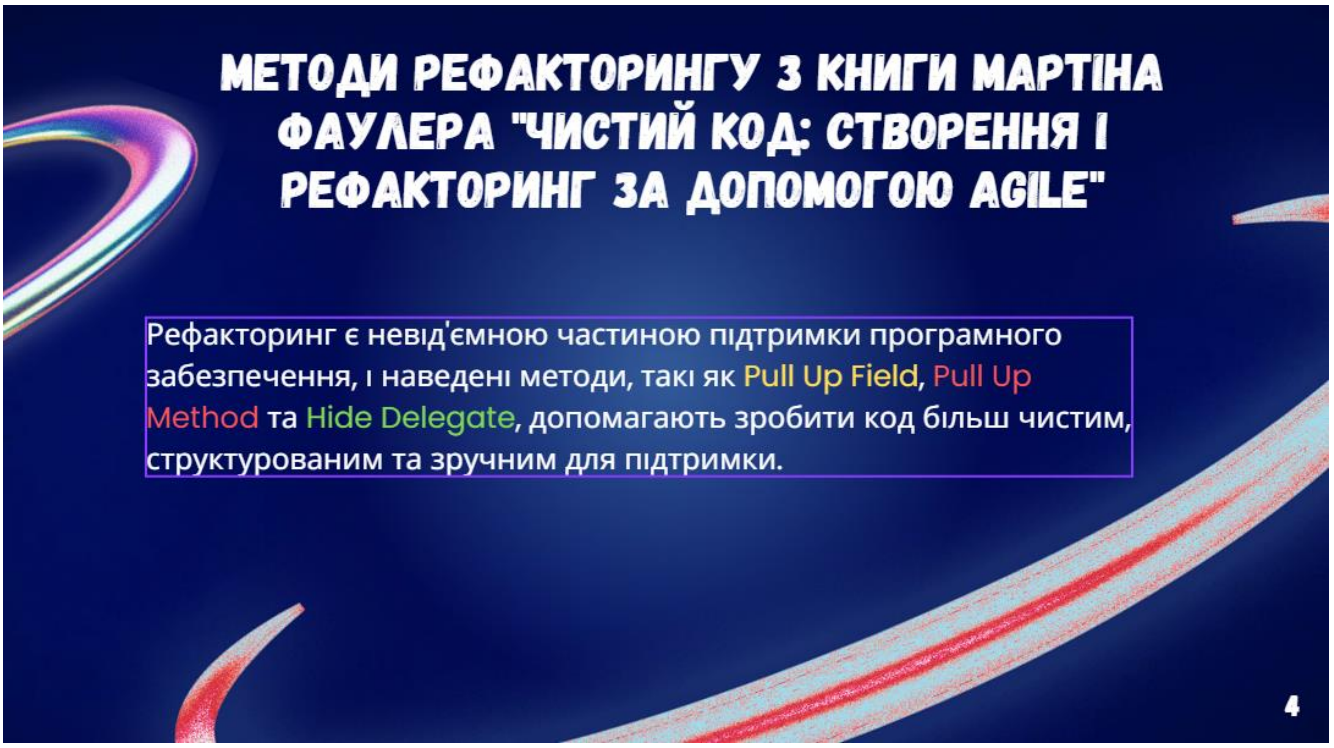
Завданням практичної роботи є написання основних рекомендацій рефакторингу коду за книгою Мартін Р. Чистий код: створення і рефакторинг за допомогою AGILE.

ВИСНОВКИ

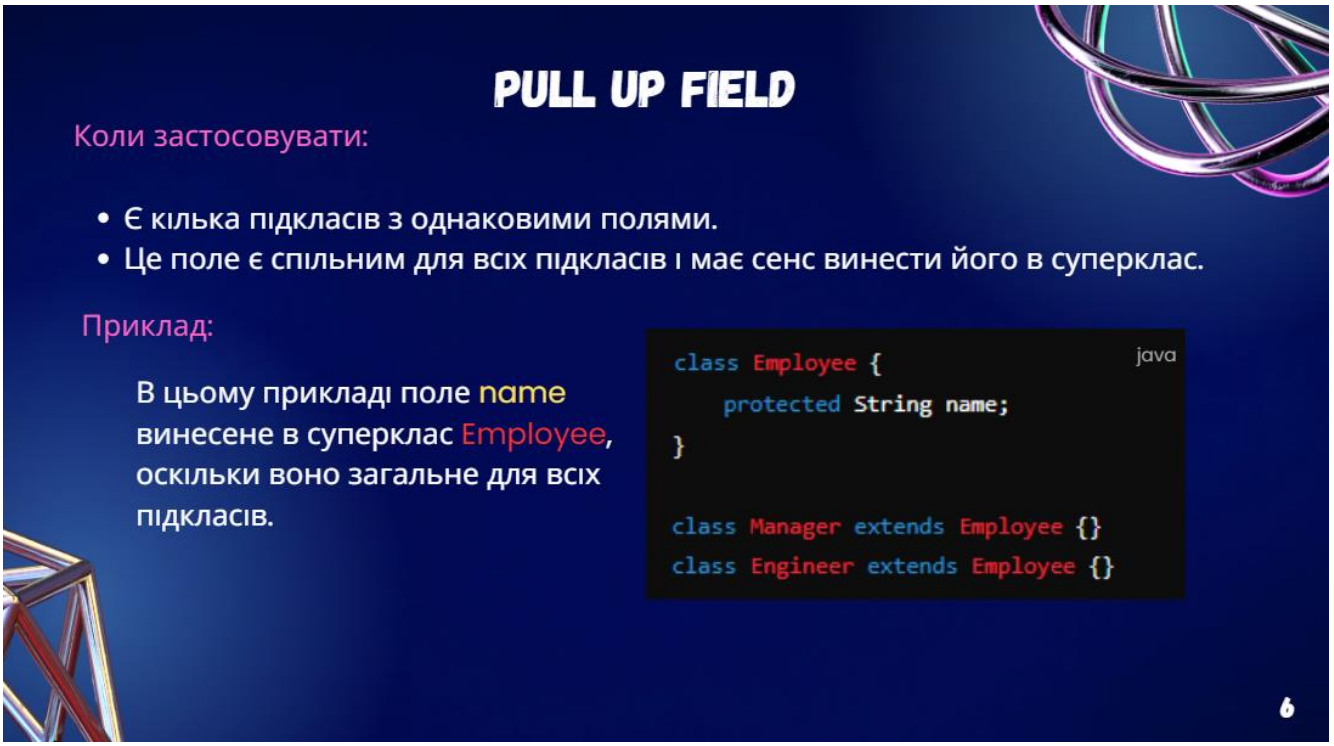
Рефакторинг є невід'ємною частиною підтримки програмного забезпечення, і наведені методи, такі як Pull Up Field, Pull Up Method та Hide Delegate, допомагають зробити код більш чистим, структурованим та зручним для підтримки.

Додаток А: Презентація


ЩО ТАКЕ РЕФАКТОРИНГ
Рефакторинг коду — це процес внесення змін до структури вихідного коду програмного забезпечення з метою покращення його читабельності, підтримки та розширення без зміни зовнішньої поведінки програми. Він не додає нові функціональні можливості, але робить код легшим для розуміння, тестування та внесення змін.
3


МЕТОДИ РЕФАКТОРИНГУ З КНИГИ МАРТІНА ФАУЛЕРА "ЧИСТИЙ КОД: СТВОРЕННЯ І РЕФАКТОРИНГ ЗА ДОПОМОГОЮ AGILE"
Рефакторинг є невід'ємною частиною підтримки програмного забезпечення, і наведені методи, такі як Pull Up Field, Pull Up Method та Hide Delegate, допомагають зробити код більш чистим, структурованим та зручним для підтримки.
4


PULL UP FIELD
(ВИТЯГУВАННЯ ПОЛЯ
В СУПЕРКЛАС)
Метод полягає в переміщенні загальних полів з кількох підкласів до суперкласу, якщо ці поля мають однакове призначення та використовуються однаковим чином.
Як це працює:
Якщо в кількох підкласах є однакове поле, його можна підняти в суперклас, щоб уникнути дублювання та зробити код більш узагальненим.
5


PULL UP FIELD
Коли застосовувати:
• Є кілька підкласів з однаковими полями.
• Це поле є спільним для всіх підкласів і має сенс винести його в суперклас.
Приклад:
В цьому прикладі поле name винесене в суперклас Employee, оскільки воно загальне для всіх підкласів.
java
class Employee {
protected String name;
}
class Manager extends Employee {}
class Engineer extends Employee {}
6

PULL UP METHOD (ВИТЯГУВАННЯ МЕТОДУ В СУПЕРКЛАС)

Цей метод використовується для переміщення загальних методів з кількох підкласів до суперкласу.

Як це працює:

Якщо однакові методи присутні в кількох підкласах, їх можна підняти в суперклас для уникнення дублювання коду.

7

PULL UP METHOD

Коли застосовувати:

- Є кілька підкласів, які мають однакові методи.
- Поведінка методу загальна для всіх підкласів, і немає потреби перевизначати його для кожного підкласу окремо.

Приклад:

Метод work() винесений в суперклас, оскільки він однаковий для всіх підкласів.

```java
class Employee {
    void work() {
        System.out.println("Working");
    }
}

class Manager extends Employee {}
class Engineer extends Employee {}
```

8

HIDE DELEGATE (ПРИХОВУВАННЯ ДЕЛЕГАТА)

Використовується для спрощення доступу до делегованого об’єкта через методи класу, що містить його.

Як це працює:

Замість того, щоб клієнтський код звертався до об’єкта через кілька рівнів делегування, ми можемо створити методи в класі-делегаті, які приховують внутрішню структуру.

9

HIDE DELEGATE

Коли застосовувати:

- Клас використовує інший клас (делегат) для виконання певних завдань.
- Клієнтський код не повинен знати про внутрішню структуру делегата.

Приклад:

Клас Person делегує виклик методу getCity() об'єкту Address, але приховує цей процес від зовнішніх класів.

```java
class Person {
    private Address address;

    public String getCity() {
        return address.getCity();
    }
}

class Address {
    private String city;

    public String getCity() {
        return city;
    }
}
```

10

ДЯКУЮ ЗА УВАГУ!

“ЧИСТИЙ КОД — ЦЕ НЕ ТІЛЬКИ ОЗНАКА ПРОФЕСІОНАЛІЗМУ, А Й ФУНДАМЕНТ СТІЙКОГО РОЗВИТКУ. РЕФАКТОРИНГ СЬОГОДНІ — ЦЕ УСПІХ ВАШИХ РІШЕНЬ ЗАВТРА.”