

Харківський національний університет радіоелектроніки
Факультет комп'ютерних наук
Кафедра програмної інженерії
ЗВІТ
з дисципліни "Аналіз та рефакторинг коду"
до практичної роботи №2
на тему: "Методи рефакторингу коду програмного забезпечення"

Виконав ст. гр. ПЗП-22-2
Д'яченко Микита Олександрович

Доц. кафедри ПІ
Лещинський Володимир Олександрович

Харків 2024

Мета роботи

Дослідити основи якості коду та застосувати практики з "Clean Code" Роберта Мартіна, а саме: значущі імена, форматування і видалення мертвого коду.

Хід роботи

Я обрав три ключові аспекти з книги "Clean Code": значущі імена (Chapter 2: Meaningful Names), форматування (Chapter 5: Formatting) і видалення мертвого коду (Dead Code, Chapter 17: Smells and Heuristics).

Значущі імена – це принцип, що передбачає використання зрозумілих та інформативних назв змінних, функцій та класів. Я застосував цей принцип, перейменувавши змінні в моєму старому проєкті, що тепер чіткіше пояснюють свою функцію в коді.

Форматування – це створення структурованого, читабельного та послідовного стилю написання коду. Я застосував це, впровадивши однакове відступлення, розбивання коду на логічні блоки та додавання коментарів для пояснення складних частин.

Видалення мертвого коду – це практика позбавлення від непотрібних фрагментів коду, які більше не використовуються або не мають сенсу. Я переглянув свій проєкт, видаливши старі методи та змінні, які не використовувалися, що зробило код компактнішим і чистішим.

Висновки:

Застосування цих практик значно підвищує якість коду, полегшує його читання та підтримку, а також сприяє розвитку кращих звичок у програмуванні.

ДОДАТОК А
Слайди презентацій

Методи рефакторингу коду програмного забезпечення

Д'яченко Микита Олександрович, ПЗПІ-22-2

Chapter 2: Meaningful Names
(Розділ 2: Значущі імена)

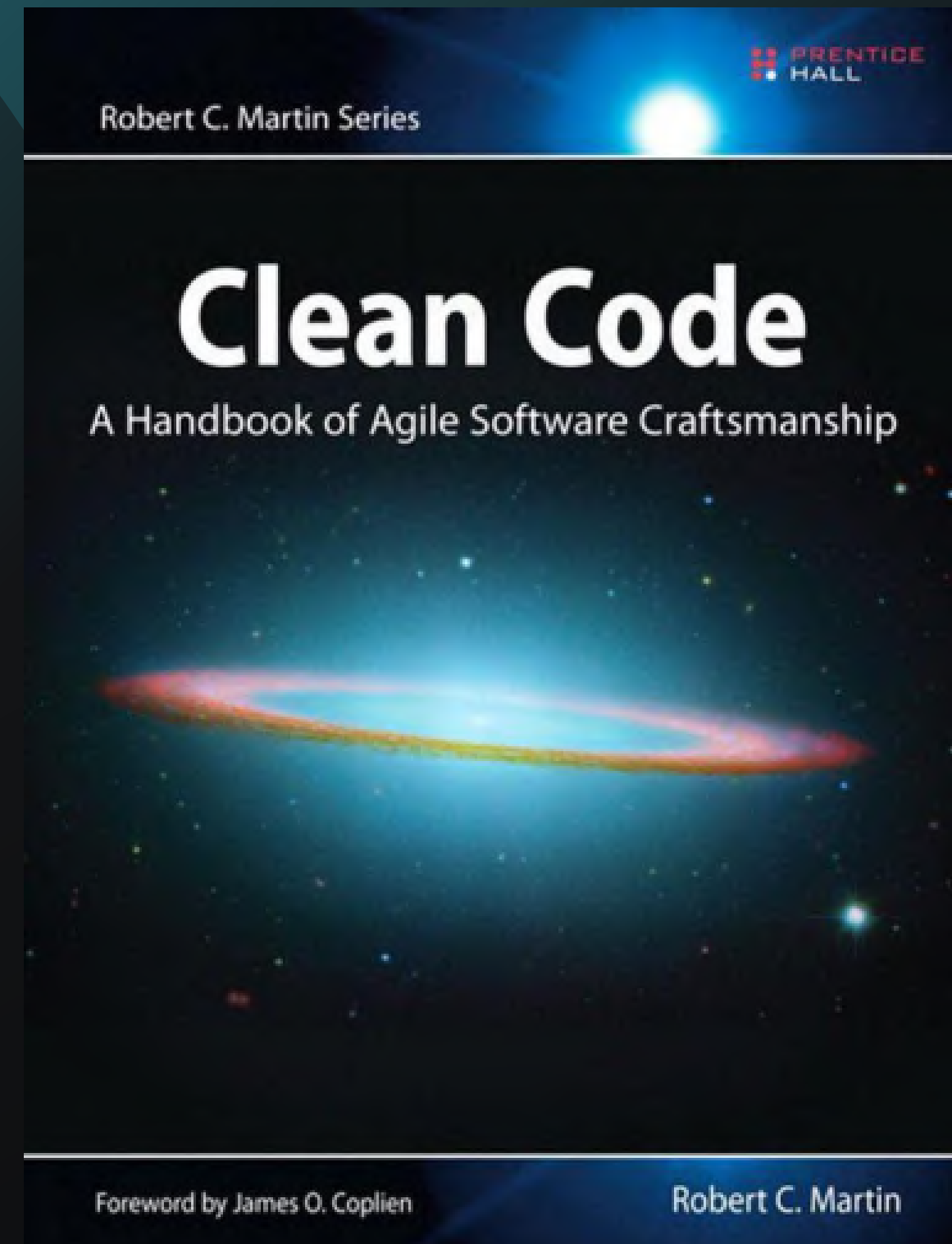
Chapter 5: Formatting
(Розділ 5: Форматування)

Dead Code, Chapter 17: Smells and Heuristics
(Мертвий код, Розділ 17: Запахи та евристика)

Джерело

Всі методи були взяті з книги
Мартіна Р. Чистий код, 2007.

Вихідник був взятий на англійській мові, тож надалі
можна буде зустріти назву методів на англійській
мові, з подальшим перекладом і поясненням.



Зміст

Chapter 2: Meaningful Names
(Розділ 2: Значущі імена)

Chapter 5: Formatting
(Розділ 5: Форматування)

Dead Code, Chapter 17: Smells and Heuristics
(Мертвий код, Розділ 17: Запахи та евристика)

The background features a series of concentric circles in shades of purple, blue, and pink. Two spheres are positioned on these circles: a teal and yellow sphere on the left, and a blue and green sphere on the right.

Що таке рефакторинг?

Рефакторинг коду — це процес покращення існуючого коду без зміни його функціональності.

Мета рефакторингу:

Підвищення читабельності.
Зменшення складності.
Покращення підтримки.

Значущі імена

Meaningful Names

Суть методу:

- Використання зрозумілих та описових імен для змінних, функцій, класів.
- Уникнення скорочень, що важко зрозуміти, та неоднозначності.

Переваги:

- Полегшує розуміння коду для команди.
- Зменшує час на виправлення помилок та внесення змін.

Імена змінних та функції тепер розкривають їхню суть, легше зрозуміти логіку програми.

```
int a = 10; // Кількість елементів
int b = 5; // Максимум
float c = 3.14; // Значення
```

```
void fn() {
    if (a > b) {
        c *= a;
    }
    std::cout << c << std::endl;
}
```

```
int elementCount = 10; // Кількість елементів
int maxLimit = 5; // Максимум
float value = 3.14; // Значення
```

```
void calculateValue() {
    if (elementCount > maxLimit) {
        value *= elementCount;
    }
    std::cout << value << std::endl;
}
```



Форматування

Formatting

Суть методу:


- Вирівнювання коду та дотримання єдиного стилю форматування.
- Використання відступів, порожніх рядків і структурованих блоків.

Переваги:

- Полегшує розуміння коду для команди.
- Зменшує час на виправлення помилок та внесення змін.

Код став більш структурованим та читабельним, додані відступи та порожні рядки.

```
if(a>10){b+=a;}else{  
b-=a;}for(int i=0;i<10;i++){std::cout<<i;}
```



```
if (a > 10) {  
    b += a;  
} else {  
    b -= a;  
}
```

```
for (int i = 0; i < 10; i++) {  
    std::cout << i << std::endl;  
}
```

Мертвий код

Dead Code

Суть методу:

- Видалення коду, який більше не використовується.
- Уникнення залишків застарілих функцій, змінних чи класів.

Переваги:

- Зменшує обсяг коду для обробки.
- Покращує продуктивність.

Видалено функції `calculateSquare` та `printHello`, які не використовуються, код став компактнішим та легшим для розуміння.

```
int calculateSquare(int x) {  
    return x * x;  
}  
  
void printHello() {  
    std::cout << "Hello!" << std::endl;  
}  
  
int calculateSum(int a, int b) {  
    return a + b;  
}  
  
int main() {  
    int result = calculateSum(5, 10);  
    std::cout << result << std::endl;  
    return 0;  
}
```

```
int calculateSum(int a, int b) {  
    return a + b;  
}  
  
int main() {  
    int result = calculateSum(5, 10);  
    std::cout << result << std::endl;  
    return 0;  
}
```



Висновки



- Рефакторинг є важливим етапом розробки програмного забезпечення.
- Використання значущих імен, форматування та видалення мертвого коду робить код чистішим.
- Якісний код сприяє зменшенню помилок і покращує ефективність роботи команди.



Дякую за увагу!