МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РАДІОЕЛЕКТРОНІКИ

Практична робота №2

з дисципліни

«Аналіз та рефакторинг коду»

на тему: «Refactoring Methods»

Виконав:

ст. гр. ПЗПІ-22-5

Коноваленко Данііл Миколайович

Перевірив:

ст. викладач каф. ПІ

Сокорчук Ігор Петрович

Харків 2024

МЕТОДИ РЕФАКТОРИНГУ КОДУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Мета роботи:

Навчитися основним методам рефакторингу коду на основі реальних прикладів з власних програмних проєктів. Навчитися ідентифікувати проблеми в коді та використовувати відповідні методи рефакторингу для покращення його якості.

Висновки:

В результаті виконання роботи була створена презентація в якій докладно були описані 3 методи рефакторингу, а саме:

1. Extract Method (Виділення методу)
2. Rename Method (Перейменування методу)
3. Replace Temp with Query (Заміна тимчасових змінних на запити)

Приклад коду був взятий з лабораторної роботи попередніх років. Рефакторинг значно покращив якість коду. Виділення методів та заміна тимчасових змінних на запити зробили його чистішим і зрозумілішим. Кожен метод виконує конкретну задачу, що підвищує ефективність підтримки та тестування коду.

Слайди презентації надані в Додатку А.

Додаток А

Методи рефакторингу коду програмного забезпечення

Коновалено Даніїл Миколайович

ПЗПІ-22-5

29.10.2024

Вступ. Рефакторинг.

- **Рефакторинг** — це процес покращення коду без зміни його зовнішньої поведінки. Рефакторинг спрямований на зміну внутрішньої структури коду для полегшення його розуміння, підвищення ефективності та забезпечення легкості подальшої підтримки й розширення.
- Основна мета рефакторингу — зробити код чистішим, більш зрозумілим і структурованим.

Основні методи рефакторингу

- Extract Method (Виділення методу)

У великому методі є кілька ділянок коду, що виконують незалежні підзадачі. Виділення методу дозволяє винести ці підзадачі в окремі методи, поліпшуючи читабельність та підтримуваність коду.

- Rename Method (Перейменування методу)

Імена методів можуть бути незрозумілими або такими, що не відображають їх реального призначення. Перейменування методу полегшує читабельність коду, оскільки нове ім'я точно описує його функцію.

- Replace Temp with Query (Заміна тимчасових змінних на запити)

У коді є тимчасові змінні для збереження результатів обчислень, що знижує його читабельність. Цей метод рефакторингу замінює ці змінні на виклик методів, які безпосередньо повертають потрібні значення.

Приклади рефакторингу на основі власного коду

1. Extract Method (Виділення методу)

До

```
function deleteTheLongestWord(str) {
    let arrSent = strSplit(str, '.');
    let allWords = [];
    let biggestWord = '';

    for (const elem in arrSent) {
        let words = strSplit(arrSent[elem], ' ');
        for (const word in words) {
            if (words[word] !== '') {
                allWords = arrPush(allWords, words[word]);
            }
        }
    }

    for (const i in allWords) {
        if (biggestWord.length < allWords[i].length) {
            biggestWord = allWords[i];
        }
    }

    allWords = filterWords(allWords, biggestWord);

    for (let i = 0; i < allWords.length; i++) {
        str = str.replace(' ' + allWords[i], '');
    }
    console.log(str);
}
```

Після

```
function deleteTheLongestWord(str) {
    let allWords = splitSentenceIntoWords(str);
    let longestWord = findLongestWord(allWords);
    allWords = filterOutLongestWord(allWords, longestWord);
    str = removeWordFromString(str, longestWord);
    console.log(str);
}

function splitSentenceIntoWords(str) {
    let sentences = str.split('.');
    let words = [];
    sentences.forEach(sentence => {
        words.push(...sentence.split(' ').filter(word => word !== ''));
    });
    return words;
}

function findLongestWord(words) {
    return words.reduce((longest, word) => word.length > longest.length ? word : longest, '')
}

function filterOutLongestWord(words, longestWord) {
    return words.filter(word => word.length !== longestWord.length);
}

function removeWordFromString(str, word) {
    const regex = new RegExp(`\\b${word}\\b`, 'g');
    return str.replace(regex, '');
}
```

1. Extract Method (Виділення методу) Пояснення

- Опис проблеми:

Функція deleteTheLongestWord виконує багато завдань: розділяє речення на слова, шукає найдовше слово, фільтрує масив, та видаляє найдовше слово з початкового рядка. Це робить функцію важкою для розуміння та підтримки.

- Застосування методу рефакторингу:

Виділяємо кілька нових методів: splitSentenceIntoWords, findLongestWord, removeWordFromString. Ці методи виконують окремі дії, полегшуючи читання коду та його модифікацію.

- Переваги:

Читабельність: Кожен метод чітко виконує одну задачу, що полегшує розуміння.

Повторне використання: Методи можна використовувати в інших частинах коду.

Легкість у тестуванні: Менші методи легше тестувати ізольовано.

Приклади рефакторингу на основі власного коду

2. Rename Method (Перейменування методу)

До

```
function arrPush(allWords, word) {
    allWords[allWords.length] = word;
    return allWords;
}

function strSplit(str, delimiter) {
    const arr = [''];
    const len = delimiter.length;
    let idx = 0;
    for (let i = 0; i < str.length; i++) {
        let sample = '';
        for (let x = i; x < i + len; x++) {
            sample += str[x];
        }
        if (sample === delimiter) {
            i += len;
            idx += 1;
            arr[idx] = '';
        }
        if (str[i])
            arr[idx] += str[i];
    }
    return arr;
}

function filterWords(allWords, biggestWord) {
    let biggestWords = [];
    for (let i = 0; i < allWords.length; i++) {
        if (allWords[i].length === biggestWord.length) {
            biggestWords[biggestWords.length] = allWords[i];
        }
    }
    return biggestWords;
}
```

Після

```
function appendWordToArray(wordsArray, word) {
    wordsArray[wordsArray.length] = word;
    return wordsArray;
}

function customSplit(str, delimiter) {
    const arr = [''];
    const len = delimiter.length;
    let idx = 0;
    for (let i = 0; i < str.length; i++) {
        let sample = '';
        for (let x = i; x < i + len; x++) {
            sample += str[x];
        }
        if (sample === delimiter) {
            i += len;
            idx += 1;
            arr[idx] = '';
        }
        if (str[i])
            arr[idx] += str[i];
    }
    return arr;
}

function filterOutLongestWord(wordsArray, longestWord) {
    let longestWords = [];
    for (let i = 0; i < wordsArray.length; i++) {
        if (wordsArray[i].length === longestWord.length) {
            longestWords.push(wordsArray[i]);
        }
    }
    return longestWords;
}
```

2. Rename Method (Перейменування методу) Пояснення

- Опис проблеми:

Імена методів arrPush, strSplit та filterWords не є інформативними, що знижує читабельність коду.

- Застосування методу рефакторингу:

Перейменовуємо методи, щоб вони краще відображали свою функціональність: arrPush на appendWordToArray, strSplit на customSplit, filterWords на filterOutLongestWord.

- Переваги:

Зрозумілість: Імена методів тепер чітко відображають їхні дії.

Легкість у підтримці: Зміна імен робить код зрозумілішим і легшим у підтримці.

7

Приклади рефакторингу на основі власного коду

3. Replace Temp with Query (Заміна тимчасових змінних на запити)

До

```
let biggestWord = '';
for (const i in allWords) {
    if (biggestWord.length < allWords[i].length) {
        biggestWord = allWords[i];
    }
}
```

Після

```
function deleteTheLongestWord(str) {
    let allWords = splitSentenceIntoWords(str);
    let longestWord = findLongestWord(allWords); // Замінили
    allWords = filterOutLongestWord(allWords, longestWord);
    str = removeWordFromString(str, longestWord);
    console.log(str);
}
```

8

3. Replace Temp with Query (Заміна тимчасових змінних на запити)
Пояснення

- Опис проблеми:

Тимчасова змінна biggestWord зберігає найдовше слово, що ускладнює структуру коду. Замінивши її на метод-запит, ми уникнемо потреби у додатковій змінній.

- Застосування методу рефакторингу:

Замінимо змінну biggestWord на виклик методу findLongestWord, який одразу поверне найдовше слово.

- Переваги:

Менше змінних: Усунення тимчасових змінних зменшує обсяг пам'яті та покращує читабельність.

Покращена структура: Код стає більш лаконічним і зрозумілим завдяки зменшенню зайвих змінних.

9

Інструменти для рефакторингу

- JetBrains IDE (IntelliJ IDEA, WebStorm, PyCharm та інші) — інструменти, що пропонують різноманітні функції рефакторингу: перейменування змінних, виділення методів, злиття чи розбиття функцій, вбудована підтримка перевірки синтаксису й аналізу коду.

- Visual Studio — середовище розробки для .NET, яке має широкий спектр інструментів для рефакторингу, включаючи можливості для роботи з великим проєктами, автоматичні підказки щодо покращення коду, виявлення дублювання коду та можливість вносити зміни в усіх місцях використання об’єкта.

- Visual Studio Code — VS Code підтримує різні розширення для рефакторингу для різних мов програмування. Завдяки підтримці плагінів, таких як Prettier, ESLint, та інші, можна автоматизувати процеси форматування та очищення коду.

- SonarQube — інструмент для аналізу якості коду, який допомагає виявити можливості для рефакторингу, знайти повторюваний код, потенційні помилки та інші вразливі місця.

10

Висновик

- Рефакторинг дозволяє зберігати код чистим, структурованим та легким для розуміння. Він забезпечує можливість для подальшого зростання проєкту та зменшує ризик появи помилок.

- Коли і як застосовувати методи рефакторингу:

1. Рефакторинг доцільно проводити після написання функціональності та проходження тестів, що гарантує збереження поведінки програми.

2. Методи рефакторингу слід застосовувати, коли в коді з'являється складність у його розумінні, підтримці або коли виникає багато дубльованого коду.

3. Регулярний рефакторинг допомагає зменшити технічний борг, підтримуючи код у придатному для розширення стані.

Список використаних джерел

1. Мартін Фаулер. Refactoring: Improving the Design of Existing Code.

2. Офіційна документація JetBrains IDE.

3. Офіційна документація Visual Studio.

4. Офіційна документація SonarQube.

5. Вебсайти та блоги, присвячені розробці програмного забезпечення: refactoring.guru, stackabuse.com.