

Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Кафедра програмної інженерії

Refactoring practices

Доповідь виконав
студент III курсу
групи ПЗП-22-2
Синенко Іван Костянтинович

Перевірив
Доц. кафедри ПІ
Лещинський Володимир Олександрович

2024

Мета роботи

Дослідити практики рефакторингу і застосувати три з них для покращення фрагментів власного коду.

Хід роботи

Я обрав три практики: Substitute Method, Replace Temp with Query, Decompose Conditional.

Substitute Method – це рефакторинг, сенс якого у заміні алгоритму, використаного у методі, на більш прозорий. Це дозволяє покращити читабельність коду. Я застосував його до одного методу, який створив свого часу для CodeWars, зменшивши кількість строчок в 4 рази і зробивши логіку видимою.

Replace Temp with Query – це заміна змінної, яка використовується лиш для зберігання значення запиту, на власне запит. Це допомагає позбутися зайвих строчок, які нічого не дають кодові. Я застосував його до частки коду, яка записувала час виконання функції у змінні. Я замінив її на звернення безпосередньо до класу Stopwatch, з якого вона брала значення.

Decompose Conditional – це рефакторинг, який значить винесення складних перевірок умов у окремий метод перевірки. Я застосував його до методу зміни назви ключового слова з мого старого курсового проекту. Я виніс в окремий метод умови і прибрав зайві перевірки.

Висновки:

Загалом всі ці методи направлені на покращення легкості читання і підтримки коду.

Іван Синенко

Рефакторинг

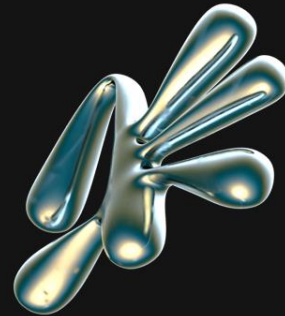
У застосуванні до мого коду



Заміна алгоритму

Substitute Algorithm

Іноді, коли в методі накопичено багато проблем, простіше переписати його наново. До того ж, заміна алгоритму допомагає коду бути більш прозорим, зрозумілим і легко підтримуваним.

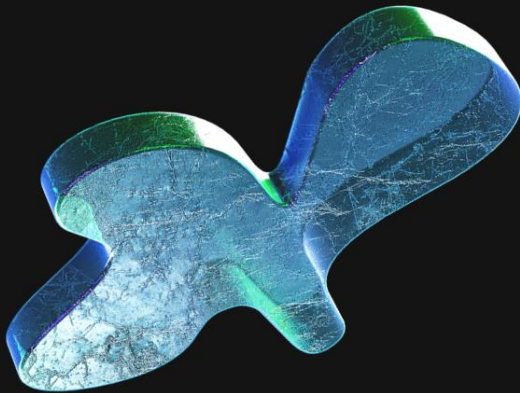


Заміна алгоритму на практиці

```
public static String whoLikesIt(String... names) {  
    //Do your magic here  
    if (names.length == 0) {  
        return "no one likes this";  
    }  
    StringBuilder sb = new StringBuilder();  
    if (names.length == 1) {  
        sb.append(names[0])  
        .append(" likes this");  
    } else if (names.length <= 3) {  
        for (int i = 0; i < names.length; i++) {  
            sb.append(names[i]);  
            if (i < names.length - 2) {  
                sb.append(", ");  
            } else if (i < names.length - 1) {  
                sb.append(" and ");  
            }  
        }  
        sb.append(" like this");  
    } else if (names.length >= 4) {  
        sb.append(names[0]).append(", ")  
        .append(names[1]).append(" and ")  
        .append(names.length - 2).append(" others ")  
        .append("like this");  
    }  
    return sb.toString();  
}
```

Як можна побачити, логіка методу стала більш прозорою, довжина зменшилась в чотири рази, він більш не потребує стрінгбілдера. І загалом, щоб розібратись у тому, що він робить, більш не потрібні сто грамів.

```
public static String whoLikesIt(String... names) {  
    switch (names.length) {  
        case 0: return "no one likes this";  
        case 1: return String.format("%s likes this", names[0]);  
        case 2: return String.format("%s and %s like this", names[0], names[1]);  
        case 3: return String.format("%s, %s and %s like this", names[0], names[1], names[2]);  
        default: return String.format("%s, %s and %d others like this", names[0], names[1], names.length - 2);  
    }  
}
```



Заміна тимчасової змінної на запит

Replace temp with query

Можна позбутись використання тимчасових змінних, що існують лише для зберігання значень.

Заміна temp на query на практиці

```
0 references
static int[,] NoClassTestWithInput(int[,] a, int[,] b, int r, int c)
{
    Console.WriteLine("TEST FOR NO CLASS\n");

    Console.WriteLine("Starting multiplying with no class...");
    Stopwatch stopwatch = new Stopwatch();
    stopwatch.Start();
    int[,] mul = NoClassMatrix.multiplyMatrices(a, b, r, c);
    stopwatch.Stop();


    long elapsedTicks = stopwatch.ElapsedTicks;
    double elapsedTime = stopwatch.Elapsed.TotalMilliseconds;

    Console.WriteLine("Multiplying with no class finished");

    Console.WriteLine($"Time spent without class: {elapsedTime} ms (Ticks: {elapsedTicks})");
    return mul;
}
```

Це метод для тестування інших методів в останній лабораторній роботі.

Код позбувся двох зайвих змінних, без яких він продовжував працювати так само.

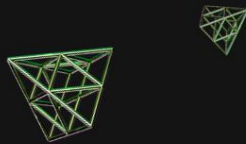


```
0 references
static int[,] NoClassTestWithInput(int[,] a, int[,] b, int r, int c)
{
    Console.WriteLine("TEST FOR NO CLASS\n");

    Console.WriteLine("Starting multiplying with no class...");
    Stopwatch stopwatch = new Stopwatch();

    stopwatch.Start();
    int[,] mul = NoClassMatrix.multiplyMatrices(a, b, r, c);
    stopwatch.Stop();

    Console.WriteLine("Multiplying with no class finished");
    Console.WriteLine($"Time spent without class: " +
        $"{stopwatch.Elapsed.TotalMilliseconds} " +
        $"{ms (Ticks: {stopwatch.ElapsedTicks})}");
    return mul;
}
```



Декомпозивання умови

Decompose conditional

При складних умовах має сенс виділяти перевірку їх усіх в окремий метод.



Декомпозивання умови на практиці

```
1 reference
private void button1_Click(object sender, EventArgs e)
{
    if (MessageBox.Show("Ви справді хочете змінити дані?", "Змінення даних", MessageBoxButtons.YesNo) == DialogResult.Yes)
    {
        try
        {
            if (new_id == null || Int32.Parse(new_id) <= 0)
            {
                if (new_keyword_name == null || new_keyword_name.Length > 36)
                {
                    using (SqlConnection sqlconn = new SqlConnection(connectionString))
                    {
                        sqlconn.Open();

                        string SQLQuery1 = $"UPDATE Keywords " +
                            $"SET keyword_name = '{new_keyword_name}' " +
                            $"WHERE keyword_id = {new_id}";

                        using (SqlCommand sqlCommand = new SqlCommand(SQLQuery1, sqlconn))
                        {
                            sqlCommand.ExecuteNonQuery();

                            Form1 form = new Form1();
                            form.Show();
                        }
                    }
                }
                else
                {
                    MessageBox.Show("Error: name must be shorter than 36 characters");
                }
            }
            else
            {
                MessageBox.Show("Error: Wrong ID");
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show($"Error: {ex.Message}");
        }
    }
}
```

Це функція змінення назв ключового слова у моєму другому курсовому. Загалом код всього проекту жахливий, страждає від повторень і відсутності наслідування.

Але у цьому випадку можливі деякі зміни:

Декомпозивання умови на практиці

(змінений код)

```
1 reference
private void button1_Click(object sender, EventArgs e)
{
    if (MessageBox.Show("Ви справді хочете змінити дані?", "Змінення даних", MessageBoxButtons.YesNo) == DialogResult.Yes)
    {
        try
        {
            ValidateArguments();
            using (SqlConnection sqlconn = new SqlConnection(connectionString))
            {
                sqlconn.Open();

                string SQLQuery1 = $"UPDATE Keywords " +
                    $"SET keyword_name = '{new_keyword_name}' " +
                    $"WHERE keyword_id = {new_id}";

                using (SqlCommand sqlCommand = new SqlCommand(SQLQuery1, sqlconn))
                {
                    sqlCommand.ExecuteNonQuery();

                    Form1 form = new Form1();
                    form.Show();
                }
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
        }
    }
}
```

Я прибрав всі перевірки у метод.

Код став виглядати значно компактніше, і до того ж, покращилася логіка обробки помилок, замість трьох варіантів тепер один.

```
1 reference
private void ValidateArguments()
{
    if (new_id == null || Int32.Parse(new_id) <= 0)
    {
        throw new Exception("Error! ID must not be null or negative");
    }
    if (new_keyword_name == null || new_keyword_name.Length > 36)
    {
        throw new Exception("Error! Keyword name must be shorter than 36 characters");
    }
}
```

Дякую за увагу!

