Министерство образования и науки Российской Федерации

Севастопольский государственный университет

Кафедра ИС

Отчет

По дисциплине: “Рефакторинг программного обеспечения”

Лабораторная работа №1

“Рефакторинг программного кода. Составление методов”

Выполнил:

ст.гр. ИС/б-17-2

Черняев Н.Г.

Проверил:

Строганов В.А.

Севастополь

2020

1 ЦЕЛЬ РАБОТЫ

Исследовать эффективность составления методов при рефакторинге программного кода. Получить практические навыки применения приемов рефакторинга методов.

2 ХОД РАБОТЫ

**1. Выделение метода.**

Код до рефакторинга:

static void Main(string[] args)

{

Console.WriteLine("Задание 1");

Console.Write("Введите размерность матрицы: ");

int size = Int32.Parse(Console.ReadLine());

Console.WriteLine("Исходный массив:");

double[,] arr = new double[size, size];

Random rnd = new Random();

for (int i = 0; i < size; i++)

for (int j = 0; j < size; j++)

arr[i, j] = rnd.NextFloat(-10, 10);

for (int i = 0, length = arr.GetLength(0); i < length; i++)

{

for (int j = 0; j < length; j++)

Console.Write(Math.Round(arr[i, j], 2) + "\t");

Console.WriteLine();

}

Console.WriteLine("Результат:");

for (int i = 0, length = arr.GetLength(0); i < length; i++)

for (int j = i + 1; j < length; j++)

{

double temp = arr[i, j];

arr[i, j] = arr[j, i];

arr[j, i] = temp;

}

for (int i = 0, length = arr.GetLength(0); i < length; i++)

{

for (int j = 0; j < length; j++)

Console.Write(Math.Round(arr[i, j], 2) + "\t");

Console.WriteLine();

}

}

В данном фрагменте кода имеется повторяющийся код для вывода матрицы, который можно вынести в отдельный метод (PrintMatrix). Также имеется два цикла для генерации и транспонирования матрицы вещественных чисел. Их также целесообразно вынести в отдельные методы, название которых поясняют их работу и увеличивает вероятность использования в других методах (GenerateMatrixOfRandomFloat и TransposeMatrix).

class Program

{

static void Main(string[] args)

{

Console.WriteLine("Задание 1");

Console.Write("Введите размерность матрицы: ");

int size = Int32.Parse(Console.ReadLine());

Console.WriteLine("Исходный массив:");

**double[,] arr = GenerateMatrixOfRandomFloat(size, size);**

**PrintArray(arr);**

Console.WriteLine("Результат:");

**TransposeMatrix(arr);**

**PrintArray(arr);**

}

**static double[,] GenerateMatrixOfRandomFloat(int sizeRow, int sizeCol)**

**{**

**double[,] arr = new double[sizeRow, sizeCol];**

**Random rnd = new Random();**

**for (int i = 0; i < sizeRow; i++)**

**{**

**for (int j = 0; j < sizeCol; j++)**

**{**

**arr[i, j] = rnd.NextFloat(-10, 10);**

**}**

**}**

**return arr;**

**}**

**static void TransposeMatrix(double[,] arr)**

**{**

**for (int i = 0, length = arr.GetLength(0); i < length; i++)**

**{**

**for (int j = i + 1; j < length; j++)**

**{**

**double temp = arr[i, j];**

**arr[i, j] = arr[j, i];**

**arr[j, i] = temp;**

**}**

**}**

**}**

**static void PrintArray(double[,] arr)**

**{**

**for (int i = 0, length = arr.GetLength(0); i < length; i++)**

**{**

**for (int j = 0; j < length; j++) {**

**Console.Write(Math.Round(arr[i, j], 2) + "\t");**

**}**

**Console.WriteLine();**

**}**

**}**

}

**2. Введение поясняющей переменной.**

Фрагмент кода до рефакторинга:

private void создатьToolStripMenuItem\_Click(object sender, EventArgs e)

{

if ((fileName == "" && textBox1.Text != "") || (fileName != "" && textBox1.Text != File.ReadAllText(fileName)))

{

DialogResult result = MessageBox.Show("Вы хотите сохранить файл?", "Сохранение", MessageBoxButtons.YesNoCancel, MessageBoxIcon.Information);

if (result == DialogResult.Yes)

Saving(fileName);

if (result == DialogResult.Cancel)

return;

}

textBox1.Text = "";

fileName = "";

filesave = false;

}

Данный фрагмент кода выполняет функцию создания файла и вызывается при клике в интерфейсе на соответствующую кнопку. В теле метода содержится сложное условие, фрагменты которого можно вынести в отдельные переменные. Ниже представлен фрагмент кода после рефакторинга:

private void создатьToolStripMenuItem\_Click(object sender, EventArgs e)

{

**bool isNewAndNotEmpty = fileName == "" && textBox1.Text != "";**

**bool isOldAndEdited = fileName != "" && textBox1.Text != File.ReadAllText(fileName);**

if (**isNewAndNotEmpty || isOldAndEdited**)

{

DialogResult result = MessageBox.Show("Вы хотите сохранить файл?", "Сохранение", MessageBoxButtons.YesNoCancel, MessageBoxIcon.Information);

if (result == DialogResult.Yes)

Saving(fileName);

if (result == DialogResult.Cancel)

return;

}

textBox1.Text = "";

fileName = "";

filesave = false;

}

**3. Встраивание метода.**

Фрагмент кода до рефакторинга:

public partial class Form1 : Form

{

...

private void fillDataHashFromMemory()

{

clearDataGridRows();

foreach (var item in hashTable.Items)

{

foreach (var value in item.Value)

{

dataGridViewHash.Rows.Add(item.Key, value.Key);

}

}

}

private void clearDataGridRows()

{

dataGridViewHash.Rows.Clear();

}

...

}

В данном фрагменте кода присутствует метод, тело которого столь же понятно, как и его название. Проведем рефакторинг, найдя все вызовы данного метода и заменив его телом метода. Код после рефакторинга представлен ниже:

public partial class Form1 : Form

{

...

private void fillDataHashFromMemory()

{

**dataGridViewHash.Rows.Clear();**

foreach (var item in hashTable.Items)

{

foreach (var value in item.Value)

{

dataGridViewHash.Rows.Add(item.Key, value.Key);

}

}

}

...

}

**4. Замещение алгоритма.**

Код метода до рефакторинга:

public static int[] TwoOldestAges(int[] ages)

{

int oldest1 = -1;

int oldest2 = -1;

foreach (int age in ages)

{

if (age > oldest1)

{

oldest2 = oldest1;

oldest1 = age;

}

else if (age > oldest2)

{

oldest2 = age;

}

}

return new int[] { oldest2, oldest1 };

}

Данная функция принимает массив возрастов и возвращает два самых больших числа в формате [второй самый старый возраст, самый старый возраст]. Данный алгоритм можно заменить более понятным если воспользоваться библиотекой Linq. Функция после рефакторинга представлена ниже:

**using System.Linq;**

public class LargestTwo

{

public static int[] TwoOldestAges(int[] ages)

{

**return ages.OrderBy(n => n).TakeLast(2).ToArray();**

}

}

**5. Замена временной переменной вызовом метода.**

Код до рефакторинга:

public partial class Form1 : Form

{

...

private void buttonAdd\_Click(object sender, EventArgs e)

{

bool textBoxIsValid = Regex.IsMatch(textBox1.Text, @"^\d+$") && textBox2.Text != "" && textBox3.Text != "" && Regex.IsMatch(textBox4.Text, @"^\d+$");

if (textBoxIsValid)

{

string[] row = { textBox1.Text, textBox2.Text, textBox3.Text, textBox4.Text };

avl.Add(Convert.ToInt32(row[0]));

avltree.Insert(Convert.ToInt32(row[0]));

var listViewItem = new ListViewItem(row);

listView.Items.Add(listViewItem);

clearFields();

saveListInFile();

}

else

{

MessageBox.Show("Ошибка в полях ввода!", "Добавление", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

addIsFail = textBoxIsValid;

}

private void buttonEdit\_Click(object sender, EventArgs e)

{

bool textBoxIsValid = Regex.IsMatch(textBox1.Text, @"^\d+$") && textBox2.Text != "" && textBox3.Text != "" && Regex.IsMatch(textBox4.Text, @"^\d+$");

if (textBoxIsValid)

{

avl.Delete(Convert.ToInt32(listView.SelectedItems[0].SubItems[0].Text));

avl.Add(Convert.ToInt32(listView.SelectedItems[0].SubItems[0].Text));

listView.SelectedItems[0].Remove();

string[] row = { textBox1.Text, textBox2.Text, textBox3.Text, textBox4.Text };

var listViewItem = new ListViewItem(row);

listView.Items.Add(listViewItem);

clearFields();

saveListInFile();

}

else

{

MessageBox.Show("Ошибка в полях ввода!", "Редактирование", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

editIsFail = textBoxIsValid;

}

...

}

В данном фрагменте кода присутствует временная переменная, которая используется для хранения значения выражения. Преобразуем выражение в метод и заменим в этих функциях переменную на вызов метода.

Код после рефакторинга:

public partial class Form1 : Form

{

...

private void buttonAdd\_Click(object sender, EventArgs e)

{

if (**validTextBoxes()**)

{

string[] row = { textBox1.Text, textBox2.Text, textBox3.Text, textBox4.Text };

avl.Add(Convert.ToInt32(row[0]));

avltree.Insert(Convert.ToInt32(row[0]));

var listViewItem = new ListViewItem(row);

listView.Items.Add(listViewItem);

clearFields();

saveListInFile();

}

else

{

MessageBox.Show("Ошибка в полях ввода!", "Добавление", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

addIsFail = **validTextBoxes()**;

}

private void buttonEdit\_Click(object sender, EventArgs e)

{

if (**validTextBoxes()**)

{

avl.Delete(Convert.ToInt32(listView.SelectedItems[0].SubItems[0].Text));

avl.Add(Convert.ToInt32(listView.SelectedItems[0].SubItems[0].Text));

listView.SelectedItems[0].Remove();

string[] row = { textBox1.Text, textBox2.Text, textBox3.Text, textBox4.Text };

var listViewItem = new ListViewItem(row);

listView.Items.Add(listViewItem);

clearFields();

saveListInFile();

}

else

{

MessageBox.Show("Ошибка в полях ввода!", "Редактирование", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

editIsFail = **validTextBoxes()**;

}

**private bool validTextBoxes()**

**{**

**bool isValid = Regex.IsMatch(textBox1.Text, @"^\d+$") && textBox2.Text != "" && textBox3.Text != "" && Regex.IsMatch(textBox4.Text, @"^\d+$");**

**return isValid;**

**}**

...

}

**6. Расщепление временной переменной.**

Код метода до рефакторинга:

public static int[] NoOdds(int[] values)

{

int temp = 0;

foreach (var v in values)

{

if (v % 2 == 0) temp++;

}

var result = new int[temp];

temp = 0;

foreach (int v in values)

{

if (v % 2 == 0) result[temp++] = v;

}

return result;

}

Данный метод предназначен для вывода нечетных значений массива и содержит временную переменную, которой неоднократно присваивается значение. Разделим эту переменную на 2 отдельные. Результат рефакторинга представлен ниже:

public static int[] NoOdds(int[] values)

{

**int evenCount = 0;**

foreach (var v in values)

{

if (v % 2 == 0) **evenCount++**;

}

var result = new int[**evenCount**];

**int counter = 0;**

foreach (int v in values)

{

if (v % 2 == 0) result[**counter++**] = v;

}

return result;

}

**7. Удаление присваиваний параметрам.**

Код программы до рефакторинга:

class Program

{

static void Main(string[] args)

{

int[] array = { 1, 5, 8, -3, -5, 19, 0, 2, 5 };

PrintSortedArray(array);

Console.ReadLine();

}

static private void PrintSortedArray(int[] array)

{

for (int i = 0; i < array.Length - 1; i++)

{

for (int j = 0; j < array.Length - i - 1; j++)

{

if (array[j + 1] < array[j])

{

int temp = array[j + 1];

array[j + 1] = array[j];

array[j] = temp;

}

}

}

for (int i = 0; i < array.Length - 1; i++)

{

Console.WriteLine(array[i]);

}

}

}

В данном коде присутствует функция, которая мутирует входящий массив, но, исходя из названия, должна только выводить отсортированный массив. Избавимся от этого, введя временную переменную. Код после изменений представлен ниже:

class Program

{

static void Main(string[] args)

{

int[] array = { 1, 5, 8, -3, -5, 19, 0, 2, 5 };

PrintSortedArray(array);

Console.ReadLine();

}

static private void PrintSortedArray(int[] array)

{

**int[] tempArray = array;**

for (int i = 0; i < **tempArray**.Length - 1; i++)

{

for (int j = 0; j < **tempArray**.Length - i - 1; j++)

{

if (**tempArray**[j + 1] < **tempArray**[j])

{

int temp = **tempArray**[j + 1];

**tempArray**[j + 1] = **tempArray**[j];

**tempArray**[j] = temp;

}

}

}

for (int i = 0; i < **tempArray**.Length - 1; i++)

{

Console.WriteLine(**tempArray**[i]);

}

}

}

ВЫВОД

В ходе выполнения лабораторной работы была исследована эффективность составления методов при рефакторинге программного кода. Получены практические навыки применения приемов рефакторинга методов.