МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

**Український державний університет науки і технологій**

Кафедра «Комп’ютерні інформаційні технології»

**Лабораторна робота №2**

**з дисципліни «Проектний практикум»**

**на тему:** «***Реалізація виділених можливостей. Розвиток проекту.»***

Виконав: студент гр. ПЗ21140 (940П)

Ахленко Д. А.

Талибов Е. Т.

Перевірив:   
Гришечкіна Т. С.

Дніпро, 2023

**Лабораторна робота №2**

**Тема: *Реалізація виділених можливостей. Розвиток проекту.***

**Мета:** Реалізувати розроблені класи, перевірити проходження всіх тестів.

1. **Постановка завдання**

Реалізація текстового редактора з можливістю рефакторингу вихідних текстів за завданням ЛР1. Моделювання зауважень замовника щодо внесення доопрацювань для наступної версії продукту.

1. **Концептуальна модель системи (узагальнена діаграма класів).**

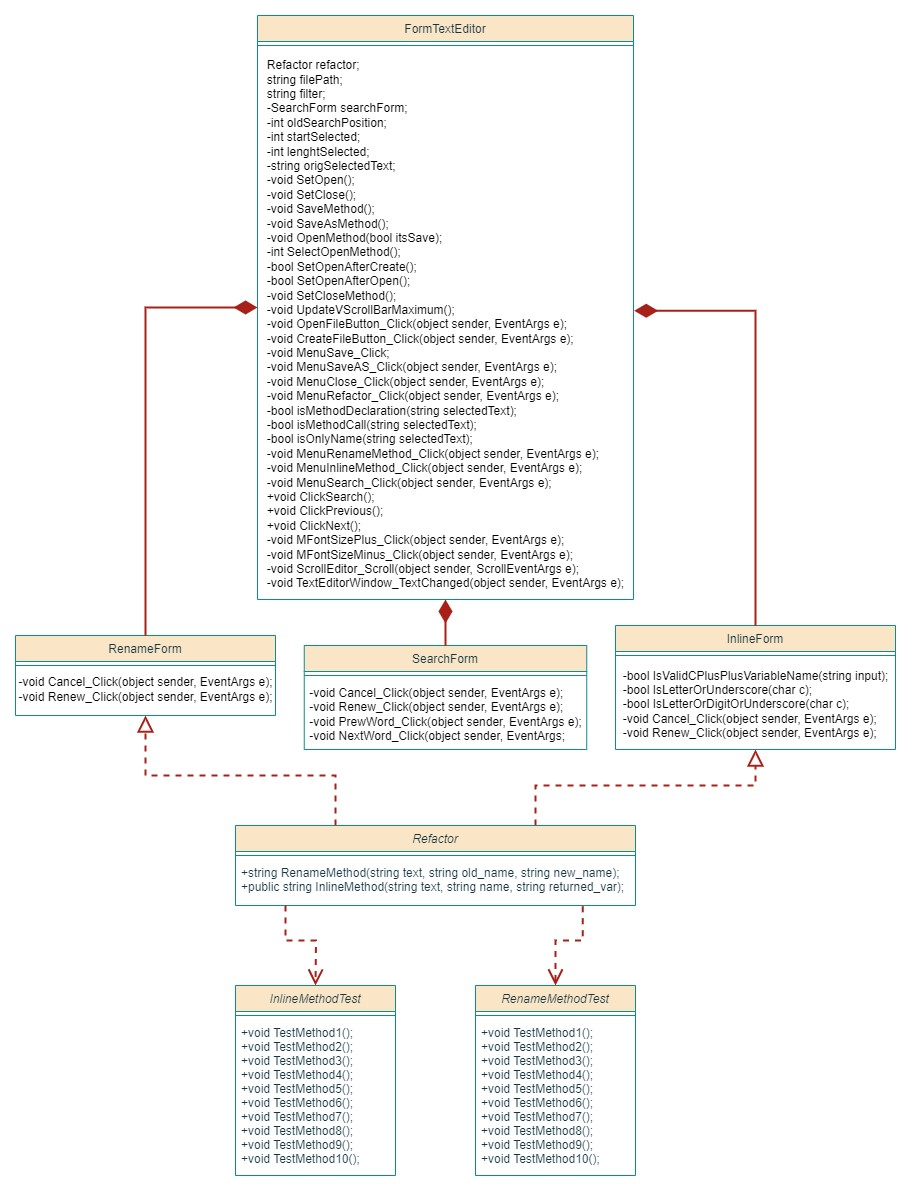


Рис. 1. – Остаточна, доопрацьована діаграма класів

1. **Текст розробленої програми**

InlineMethod:

public string InlineMethod(string text, int selectionStart, string name, string returned\_var) {

string pattern = @"\(([^)]\*)\)"; string Variables = "";

MatchCollection matches = Regex.Matches(name, pattern);

if (matches.Count > 0) Variables = matches[0].Groups[1].Value;

string OriginalClassName = name.Substring(0, name.IndexOf('.'));

if (!text.Contains("class " + OriginalClassName)) {

LinkedList<string> classNames = new LinkedList<string>();

int searchIndex = 0; string subStr = text;

bool terminate = false; while (!terminate) {

int index = subStr.IndexOf("class");

if (index != -1) { string nameCl = ""; int i = 0;

subStr = subStr.Substring(index + "class ".Length);

while (subStr[i] != ' ' && subStr[i] != '{') { nameCl += subStr[i]; i++; }

classNames.AddLast(nameCl); searchIndex += 1;

} else terminate = true;

} if (searchIndex == 0) return "Worning! Not allowed to inline method." +

"\n Methot description not found";

else { foreach (string lines in classNames) {

if (text.Contains(lines + " " + OriginalClassName)) {

OriginalClassName = lines; break; }

} if (OriginalClassName == name.Substring(0, name.IndexOf('.')))

return "Worning! Not allowed to inline method." +

"\n Methot description not found";

}

}

int id; string ClassDeclaratoin = "class " + OriginalClassName;

int index2 = text.IndexOf(ClassDeclaratoin);

id = index2 += ClassDeclaratoin.Length;

do { ClassDeclaratoin += text[id]; id++; }

while (text[id] != '{'); ClassDeclaratoin += '{';

int cordsLevel = 1; while (cordsLevel != 0) {

id++; if (text[id] == '{') cordsLevel++;

if (text[id] == '}') cordsLevel--;

ClassDeclaratoin += text[id];}

string MethodName = ""; index2 = name.IndexOf('.') + 1;

while (name[index2] != '(') { MethodName += name[index2]; index2++;}

string InsertedMethodText = "";

id = ClassDeclaratoin.IndexOf(MethodName);

do { id++; } while (ClassDeclaratoin[id] != '{');

cordsLevel = 1; while (cordsLevel != 0){

id++; if (ClassDeclaratoin[id] == '{') cordsLevel++;

if (ClassDeclaratoin[id] == '}') cordsLevel--;

InsertedMethodText += ClassDeclaratoin[id];

} InsertedMethodText = InsertedMethodText.Remove(

InsertedMethodText.Length - 1);

string MethodVariables = "";

id = ClassDeclaratoin.IndexOf(MethodName);

do { id++; } while (ClassDeclaratoin[id] != '(');

cordsLevel = 1; while (cordsLevel != 0) {

id++; if (ClassDeclaratoin[id] == '(') cordsLevel++;

if (ClassDeclaratoin[id] == ')') cordsLevel--;

MethodVariables += ClassDeclaratoin[id];

} MethodVariables = MethodVariables.Remove(

MethodVariables.Length - 1);

string ReturnMethodType = "";

id = ClassDeclaratoin.IndexOf(MethodName);

do { id--; } while (ClassDeclaratoin[id] != ' '); id--;

while (ClassDeclaratoin[id] != ' ') { id--;} id++;

while (ClassDeclaratoin[id] != ' ') {

ReturnMethodType += ClassDeclaratoin[id]; id++;}

string[] segMetVar = MethodVariables.Split(',');

string[] segVar = Variables.Split(',');

id = 0; foreach (string s in segMetVar) {

if (s.Length > 0) {

segVar[id] = segVar[id].Trim(); string st = s.Trim();

string[] temp = st.Split(' '); segMetVar[id] = temp[temp.Length - 1]; id++;

}

} id = 0; foreach (string s in segMetVar) {

if (s.Length > 0)

InsertedMethodText = InsertedMethodText.Replace(s, segVar[id]); id++;

}

if (returned\_var != "\_")

InsertedMethodText = InsertedMethodText.Replace("return ", returned\_var + " = ");

else

InsertedMethodText = InsertedMethodText.Replace("return", "");

string[] Inserted = InsertedMethodText.Split('\r');

foreach (string s in Inserted) {

id = Array.IndexOf(Inserted, s);

if (s.Contains("\n")) Inserted[id] = s.Replace("\n", "");

Inserted[id] = Inserted[id].Trim();}

string ReturnedCode = "//MInline: " + name;

if (returned\_var != "\_")

ReturnedCode += "\r\n" + ReturnMethodType + " " + returned\_var + "; {\r\n";

else ReturnedCode += "{\r\n";

foreach (string s in Inserted)

if (s.Length != 0) ReturnedCode += " " + s + "\r\n";

ReturnedCode += "}\r\n";

string LineWithMethodColl = "";

int spaces = 0; id = selectionStart;

while (text[id] != '\n') { id--; } id++;

while (text[id] == ' ') { id++; spaces++; }

while (text[id] != '\r') { LineWithMethodColl += text[id]; id++; }

string line = LineWithMethodColl;

if (returned\_var != "\_")

LineWithMethodColl = LineWithMethodColl.Replace(name, returned\_var);

else LineWithMethodColl = LineWithMethodColl.Replace(name + ";\r\n", "");

string sp = ""; while (spaces != 0) { sp += " "; spaces--; };

ReturnedCode = ReturnedCode.Replace("\n", "\n" + sp);

ReturnedCode += LineWithMethodColl;

ReturnedCode = text.Replace(line, ReturnedCode);

return ReturnedCode;

}

RenameMethod:

public string RenameMethod(string text, string old\_name, string new\_name, bool comment) {

// Розділимо текст на рядки для обробки кожного рядка окремо

string[] lines = text.Split('\n');

List<string> modifiedLines = new List<string>();

bool isMultiLineComment = false; // Прапор, що вказує на багаторядковий коментар

foreach (string line in lines) {

// Перевіримо, чи рядок починається з багаторядкового коментаря "/\*"

if (line.Trim().StartsWith("/\*")) isMultiLineComment = true;

// Перевіримо, чи рядок закінчується багаторядковим коментарем "\*/"

if (line.Trim().EndsWith("\*/")) isMultiLineComment = false;

// Якщо рядок не є багаторядковим коментарем, то ми можемо перевірити, чи містить він оголошення методу

if (!isMultiLineComment) {

// Перевіримо, чи містить рядок оголошення методу зі старою назвою

if (line.Contains(old\_name + "(")) {

if (line.Trim().StartsWith("//")) {//Перевіримо, чи є рядок однорядковим коментарем

if (comment) {//Якщо чекбокс відмічений

// Замінимо стару назву на нову в рядку

string modifiedLine = line.Replace(old\_name, new\_name);

modifiedLines.Add(modifiedLine);

} else { // Якщо чекбокс не відімчений - залишаємо однорядковий коментар без змін

modifiedLines.Add(line);

}

} else {

// Замінимо стару назву на нову в рядку

string modifiedLine = line.Replace(old\_name, new\_name);

modifiedLines.Add(modifiedLine);

}

} else {

// Якщо рядок не містить оголошення методу зі старою назвою, залишимо його без змін

modifiedLines.Add(line);

}

} else { // Якщо рядок є багаторядковим коментарем

if (comment) { //відмічений чекбокс-замінюємо стару назву

if (line.Contains(old\_name + "(")) {

// Замінимо стару назву на нову в рядку

string modifiedLine = line.Replace(old\_name, new\_name);

modifiedLines.Add(modifiedLine);

} else {

// Якщо рядок не містить оголошення методу зі старою назвою, залишимо його без змін

modifiedLines.Add(line);

}

} else { //то додаємо його без змін

modifiedLines.Add(line);

}

}

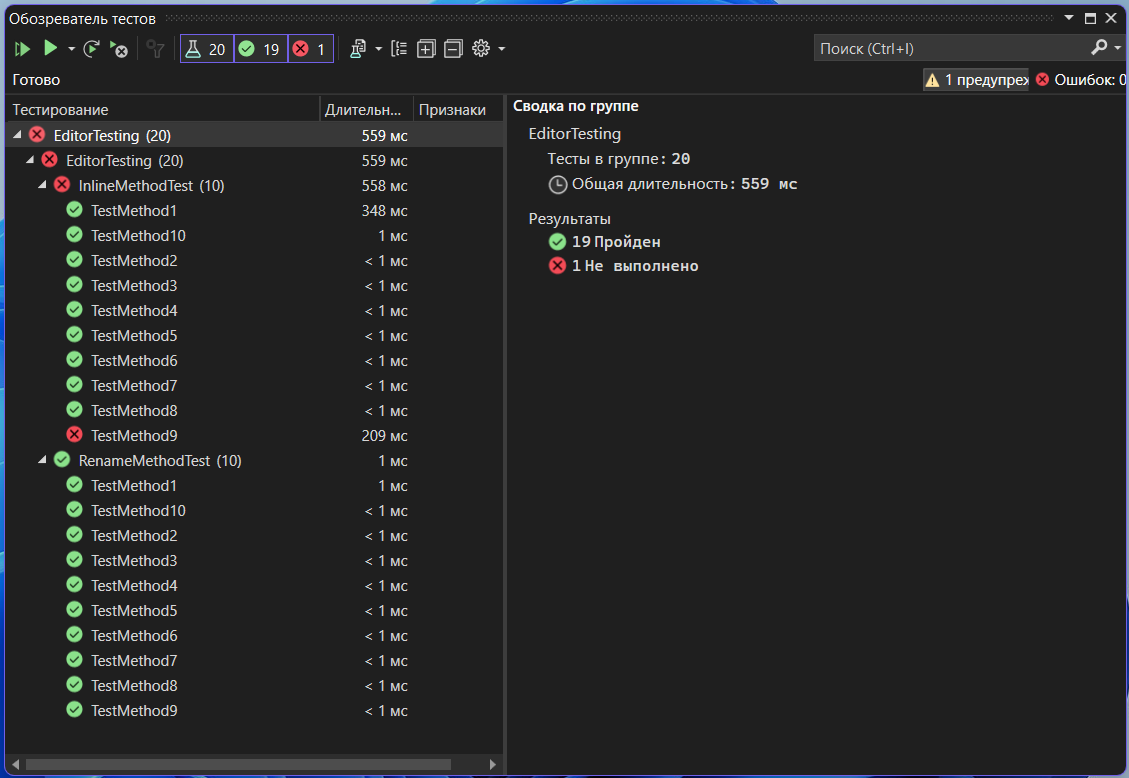
} // Об'єднаємо модифіковані рядки назад у текст коду

string modifiedText = string.Join("\n", modifiedLines);

return modifiedText;

}

1. **Проходження складених тестів**

****

Можна побачити що усі unit-тести окрім 9-го виконалися. Цей тест не було успішно пройдено, оскільки у реалізації методу рефакторингу не враховано можливість вбудови методу всередині класу що його й реалізує.

1. **Опис побажань щодо покращення функціональних можливостей системи.**

Враховуючи той факт що програмний продукт розроблений під час виконання завдання лабораторної роботи складається з трьох основних компонентів: інтерфейсу користувача, функцій рефакторингу та функції відповідної за визначення припустимих для виділеного фрагменту оброблюваного тексту програми функцій рефакторингу. Розглянемо недоліки функціоналу раніше наведених складових та висунемо декілька пропозицій щодо вдосконалення кожної частини програмного продукту:

1. **Інтерфейс користувача та його функціональна складова:**

Головною задачею інтерфейсу користувача при виконанні задач обробки програмного коду (рефакторингу) є зручний перегляд програмного коду, його редагування та навігація кодом програми.

В межах програми функціонал інтерфейсу щодо навігації реалізовано за допомогою елементу прокрутки що забезпечує переміщення по програмному коду, але, даний елемент жодним чином не реагує на прокручування коліщатка миші, що ускладнює навігацію користувача та змушує його до постійної прямої взаємодії з елементом прокрутки текстового поля.

Також, візуально програмний код відображуваний у текстовому полі має однотонне чорне забарвлення та жодні його елементи не виділяються кольорами, звичайно, забезпечення редактору підсвіткою коду вимагає великих зусиль щодо його реалізації та не є обов’язковим до реалізації в межах завдання лабораторної роботи

1. **Функції рефакторингу коду С++:**

* *Перейменування методу (Rename Method):*

В реалізованій програмі забезпечено перейменування методів що мають конкретне ім’я, також забезпечено можливість перейменування закоментованих викликів методів класу. Але, слід зазначити що реалізація методу рефакторингу щодо перейменування методу не передбачає можливості наявності в межах програми декількох класів що реалізують метод з однаковим іменем.

З точки зору реалізації перейменування конкретного методу саме конкретного класу є складним для вирішення завданням та вимагає від імені методу що слід перейменувати визначити конкретний клас у котрому його реалізовано. Також, дане завдання є досить складним оскільки оголошення класу до якого викликано метод може виконуватись як напряму Class class, так і з використанням структур даних або вкладеності класів, що значно ускладнить реалізацію методу перейменування, та, скоріш за все, вимагатиме використання рекурсивних підходів, або ж лексичного-граматичного аналізу.

* *Вбудова методу (Inline Method):*

До методу що відповідає за вбудову методу можна було б винести аналогічні припущення, але під час реалізації необхідність визначення конкретного класу котрому належить метод була врахована, також було враховано перевірку наявності реалізації самого методу в межах опису класу якому він належить, але, це в свою чергу не гарантує правильної роботи методу рефакторингу *Inline Method* у багатьох ситуаціях, наприклад якщо код методу що вбудовується містить виклики методів інших класів, але ці методи є приватними, або ж якщо клас що реалізує метод оголошено у програмі не напряму а з використанням структур даних.

1. **Визначення допустимого виду рефакторингу:**

Оскільки в межах програмного продукту реалізовано визначення припустимого для фрагменту коду методу рефакторингу слід зазначити, що при розширення функціональності (додаванні нових методів рефакторингу) дану функцію потрібно буде адаптовувати під нові додані методи рефакторингу оскільки при її написанні враховано лише функціонал щодо *Перейменування та вбудови методу.*

1. **Аналіз результатів.**

У ході виконання даної лабораторної роботи було досягнуто головної мети - реалізації виділених можливостей в проекті текстового редактора. Під час виконання завдання були реалізовані класи, які дозволяють виконувати рефакторинг вихідних текстів, враховуючи завдання ЛР1.

У ході роботи було ретельно перевірено проходження всіх тестів, що були розроблені для перевірки роботи редактора. Всі тести пройшли успішно, крім одного, що підтверджує коректну реалізацію функціональності та відсутність критичних помилок.

Також були враховані зауваження замовника щодо внесення доопрацювань для наступної версії продукту. Реалізовані можливості враховують вимоги та очікування замовника, що сприятиме подальшому розвитку проекту та задоволенню потреб користувачів.

Отже, результати цієї лабораторної роботи свідчать про успішну імплементацію важливих функцій та готовність проекту для подальшого розвитку та вдосконалення.