

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

**Український державний університет науки і технологій**

Кафедра «Комп’ютерні інформаційні технології»

**Лабораторна робота №3**

**з дисципліни «Проектний практикум»**

**на тему:**

**«Інкрементна розробка програм. Створення наступної версії продукту.»**

Виконав студент групи ПЗ2011:

Кулик С. В.

Прийняв:

Доц. Іванов О. П.

Дніпро, 2023

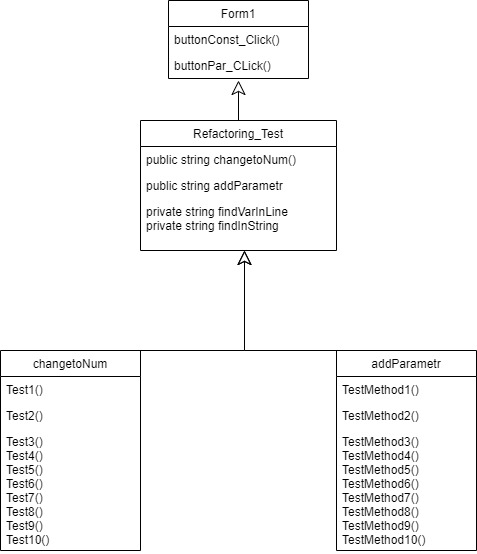
Тема: Інкрементна розробка програм. Створення наступної версії продукту.

1. **Завдання**

Реалізація текстового редактора з можливістю рефакторингу вихідних текстів за завданням ЛР1. Аналіз отриманих зауважень замовника, виконання наступної ітерації розробки проекту, отримання нової версії продукту із функціональністю двох бригад.

Об’єднання реалізацій між двома бригадами: поточна бригада використовує реалізацію наступної за номером бригади, остання за номером використовує розробку першої.

1. **Діаграма класів**

****

1. **Код програми**

**3.1. Program.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp1

{

public class Refactor\_Test

{

public string changetoNum(string text, string value, string Newvalue, bool inString = false, bool inComment = false)

{

string[] line = text.Split(new char[] { '\"' });

string result = "";

for (int i = 0; i < line.Length; i++)

{

if (i % 2 == 1)

{

if (inString)

{

result += "\"";

result += findInString(line[i], value, Newvalue);

result += "\"";

}

else

{

result += "\"";

result += line[i];

result += "\"";

}

}

else { result += findVarInLine(line[i], value, Newvalue, inComment); }

}

return result;

}

public string addParametr(string text, string name, string newName, bool inString = false, bool inComment = false)

{

string[] line = text.Split(new char[] { '\"' });

string result = "";

for (int i = 0; i < line.Length; i++)

{

if (i % 2 == 1)

{

if (inString)

{

result += "\"";

result += findInString(line[i], name, newName);

result += "\"";

}

else

{

result += "\"";

result += line[i];

result += "\"";

}

}

else { result += findVarInLine(line[i], name, newName, inComment); }

}

return result;

}

private string SearchConst(string lines, string name, string newName)

{

string result = "";

bool isEnd = false;

while (!isEnd)

{

int index = lines.IndexOf(name);

if (index >= 0)

{

if (lines[index + name.Length] == ' ' || lines[index + name.Length] == ';' || lines[index + name.Length] == ')'

|| lines[index + name.Length] == '.' || lines[index + name.Length] == ',')

{

result += lines.Substring(0, index);

result += newName;

lines = lines.Substring(index + name.Length);

}

else

{

result += lines.Substring(0, index + 1);

lines = lines.Substring(index + 1);

}

}

else

{

result += lines;

isEnd = true;

}

}

return result;

}

private string findVarInLine(string text, string name, string newName, bool inComent)

{

string result = "";

bool isEnd = false;

while (!isEnd)

{

int index = text.IndexOf("//");

if (index >= 0)

{

result += SearchConst(text.Substring(0, index), name, newName);

int indexEndComent = text.Substring(index).IndexOf("\n");

if (indexEndComent >= 0)

{

if (inComent)

{

result += findInString(text.Substring(index, indexEndComent), name, newName);

}

else

{

result += text.Substring(index, indexEndComent);

}

text = text.Substring(index + indexEndComent);

}

else

{

if (inComent)

{

result += findInString(text.Substring(index), name, newName);

}

else

{

result += text.Substring(index);

}

isEnd = true;

}

}

else

{

result += SearchConst(text, name, newName);

isEnd = true;

}

}

return result;

}

private string findInString(string text, string name, string newName)

{

string result = "";

bool isEnd = false;

while (!isEnd)

{

int index = text.IndexOf(name);

if (index >= 0 && text[index + name.Length] == ' ')

{

result += text.Substring(0, index);

result += newName;

text = text.Substring(index + name.Length);

}

else

{

result += text;

isEnd = true;

}

}

return result;

}

public string extract(string text, string textExtract, string methodName, bool extractComments = false)

{

if (!extractComments && !text.Contains(textExtract))

return text;

if (extractComments)

{

string[] textExtractCommnets = textExtract.Split(new char[] { '\n' });

string[] startTrim = new string[textExtractCommnets.Length];

string temp;

for (int i = 0; i < textExtractCommnets.Length; i++)

{

if (textExtractCommnets[i] == "")

continue;

temp = textExtractCommnets[i];

textExtractCommnets[i] = textExtractCommnets[i].TrimStart(new char[] { '\t' });

startTrim[i] = temp.Replace(textExtractCommnets[i], "");

textExtractCommnets[i] = String.Concat($"{startTrim[i]}//", textExtractCommnets[i], "\n");

}

string textExtractCommet = String.Concat(textExtractCommnets);

if (!text.Contains(textExtractCommet) && !text.Contains(textExtract))

return text;

text = text.Replace(textExtractCommet, $"\t//{methodName}();\n");

}

text = text.Replace(textExtract, $"\t{methodName}();\n");

string newMethod =

"\nvoid " + methodName + "()\n" +

"{\n" +

textExtract +

"}";

text = String.Concat(text, newMethod);

return text;

}

}

}

3.2. Form1.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace ConsoleApp1

{

public partial class Form1 : Form

{

private string fileName;

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new Form1());

}

public Form1()

{

InitializeComponent();

}

private void Form1\_Load(object sender, EventArgs e)

{

}

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

{

string name = Convert.ToString(richTextBox1.Text);

string new\_name = Convert.ToString(textBoxPar.Text);

Refactor\_Test r = new Refactor\_Test();

bool inComent = checkBoxSearchInComment.Checked;

bool inString = checkBoxSearchInString.Checked;

string text = Convert.ToString(richTextBox1.Text)+Convert.ToString(textBoxPar.Text);

string rez = r.addParametr(text, name, new\_name, inString, inComent);

richTextBox1.Text = Convert.ToString(rez);

}

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

{

string new\_name = Convert.ToString(textBoxConst.Text);

string name = Convert.ToString(textBoxNum.Text);

string text = Convert.ToString(richTextBox1.Text);

Refactor\_Test r = new Refactor\_Test();

bool inComent = checkBoxSearchInComment.Checked;

bool inString = checkBoxSearchInString.Checked;

string rez = r.addParametr(text, name, new\_name, inString, inComent);

richTextBox1.Text = Convert.ToString(rez);

}

private void exampleToolStripMenuItem1\_Click(object sender, EventArgs e)

{

richTextBox1.Text = @"package test.classification.imagenet;

public class TestVGG {

public static void main(String[] args) throws Exception {

NeuralNetwork vgg16 = new VGG()

.addInputLayer(64, 64, 3)

.addConvolutionLayer(32, 3)

.addConvolutionLayer(32, 3)

.addMaxPoolingLayer()

.addConvolutionLayer(64, 3)

.addConvolutionLayer(64, 3)

.addMaxPoolingLayer()

.addConvolutionLayer(128, 3)

.addConvolutionLayer(128, 3)

.addConvolutionLayer(128, 3)

.addMaxPoolingLayer()

.addConvolutionLayer(256, 3)

.addConvolutionLayer(256, 3)

.addConvolutionLayer(256, 3)

.addMaxPoolingLayer()

.addConvolutionLayer(512, 3)

.addConvolutionLayer(512, 3)

.addConvolutionLayer(512, 3)

.addMaxPoolingLayer()

.addDenseLayer(1024, new FunctionActivation.ReLU())

.addDropoutLayer(0.25)

.addDenseLayer(1024, new FunctionActivation.ReLU())

.addDropoutLayer(0.25)

.addDenseLayer(200, new FunctionActivation.Softmax())

.createVGG()

.setFunctionLoss(new FunctionLoss.CrossEntropy())

.setOptimizer(new AdamOptimizer())

.create();

TinyImageNetLoader3D loader = new TinyImageNetLoader3D(new TransformData.Tanh());

vgg16.info();

DataTrainer trainer = new DataTrainer(1000, 100, loader);

for (int i = 0; i < 100; i++)

{

long start = System.nanoTime();

trainer.train(vgg16, 60, 1, new DataMetric.Top1());

trainer.score(vgg16, new DataMetric.Top1());

System.out.println((System.nanoTime() - start) / 1000000);

}

}

void function(a){

int x = 0;

}

}";

}

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

{

richTextBox1.Text = "";

}

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

{

MessageBox.Show("Програма створена з метою навчання та предназначена для редагування набраного на клавіатурі тексту або тексту з файлу.", "Про програму", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

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

{

DialogResult dialogResult = MessageBox.Show("Бажаєте вийти?", "Вихід", MessageBoxButtons.YesNo, MessageBoxIcon.Question);

if (dialogResult == DialogResult.Yes)

{

System.Windows.Forms.Application.Exit();

}

}

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

{

OpenFileDialog openFileDialog1 = new OpenFileDialog();

if (openFileDialog1.ShowDialog() == DialogResult.Cancel)

return;

fileName = openFileDialog1.FileName;

saveToolStripMenuItem1.Enabled = true;

richTextBox1.Text = File.ReadAllText(fileName);

}

private void saveToolStripMenuItem1\_Click(object sender, EventArgs e)

{

if (fileName != "")

{

System.IO.FileStream fs = new System.IO.FileStream(fileName, FileMode.Open);

byte[] info = new UTF8Encoding(true).GetBytes(richTextBox1.Text);

fs.Write(info, 0, info.Length);

fs.Close();

}

}

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

{

string extraxtText = Convert.ToString(richTextBox1.SelectedText);

string name = Convert.ToString(textBoxMethodName.Text);

string text = Convert.ToString(richTextBox1.Text);

Refactor\_Test r = new Refactor\_Test();

bool inComent = checkBoxExtract.Checked;

string rez = r.extract(text, extraxtText,name, inComent);

richTextBox1.Text = Convert.ToString(rez);

}

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

{

DialogResult dialogResult = MessageBox.Show("Бажаєте сторити новий файл?", "Створити", MessageBoxButtons.YesNo, MessageBoxIcon.Question);

if (dialogResult == DialogResult.Yes)

{

richTextBox1.Text = "";

fileName = "";

}

}

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

{

SaveFileDialog saveFileDialog1 = new SaveFileDialog();

saveFileDialog1.Filter = "Text|\*.txt|C#|\*.cs|C++|\*.cpp";

saveFileDialog1.Title = "Збереження";

saveFileDialog1.ShowDialog();

if (saveFileDialog1.FileName != "")

{

fileName = saveFileDialog1.FileName;

saveToolStripMenuItem1.Enabled = true;

System.IO.FileStream fs = (System.IO.FileStream)saveFileDialog1.OpenFile();

byte[] info = new UTF8Encoding(true).GetBytes(richTextBox1.Text);

fs.Write(info, 0, info.Length);

fs.Close();

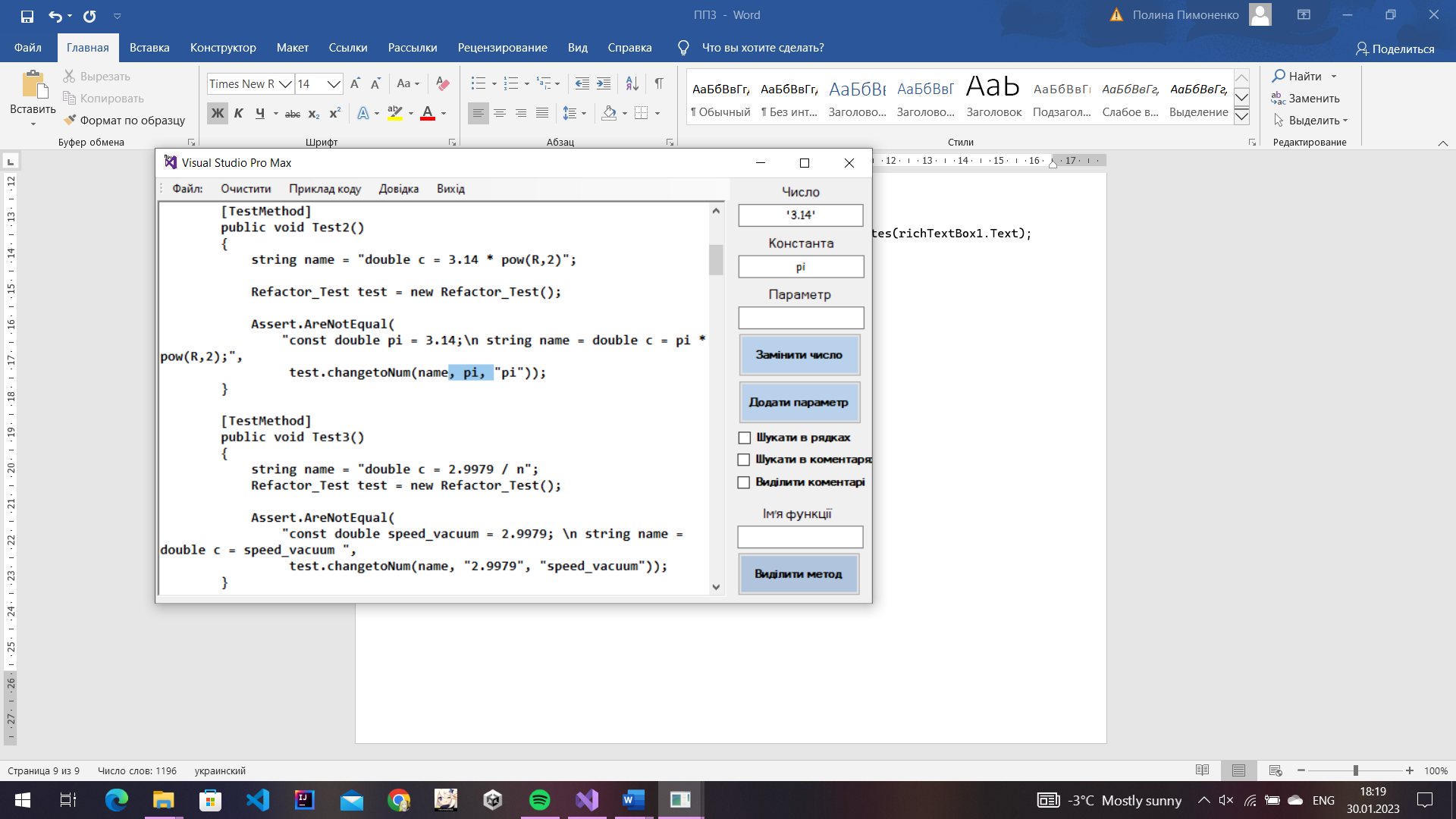
}

}

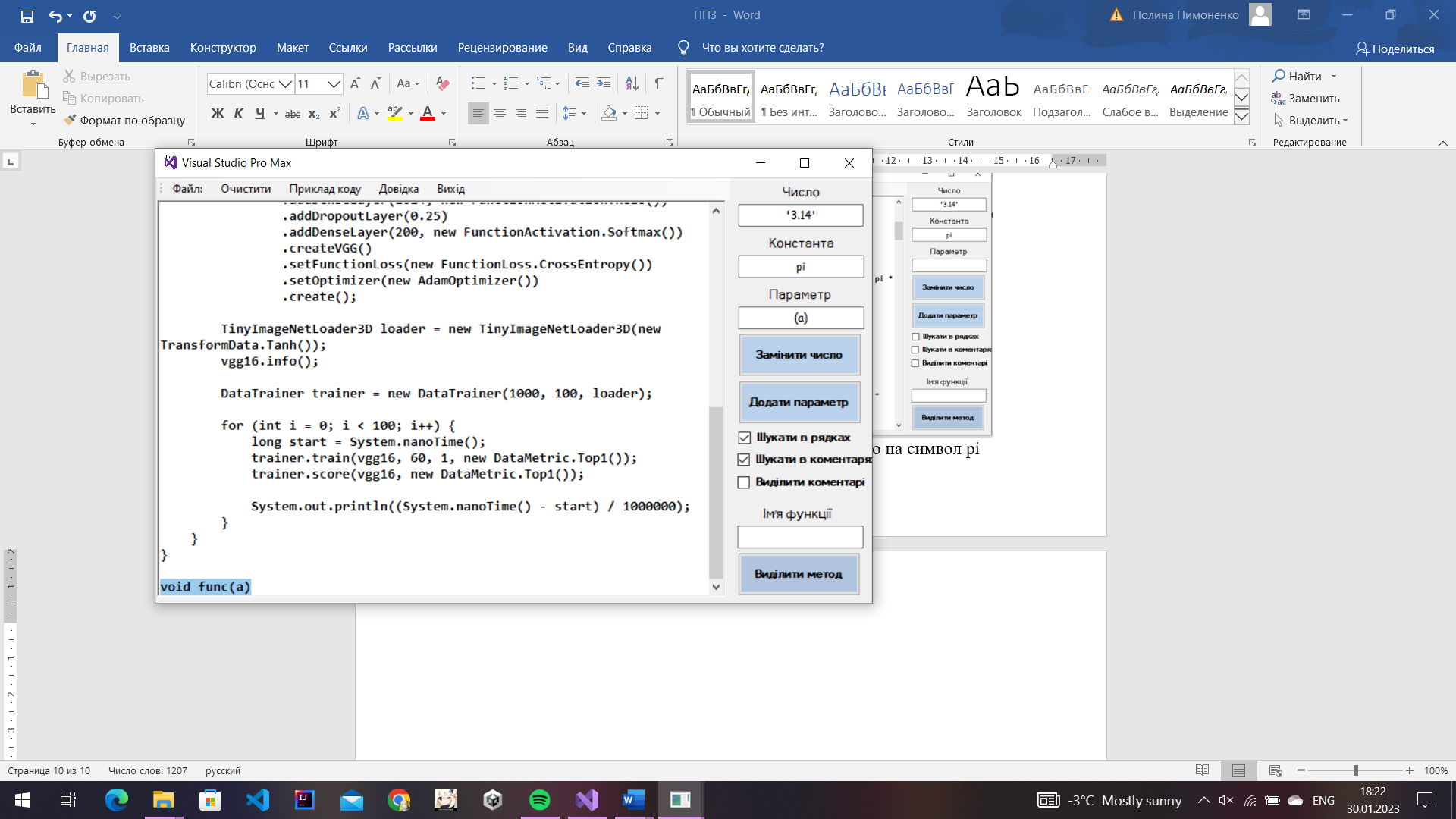
}

}

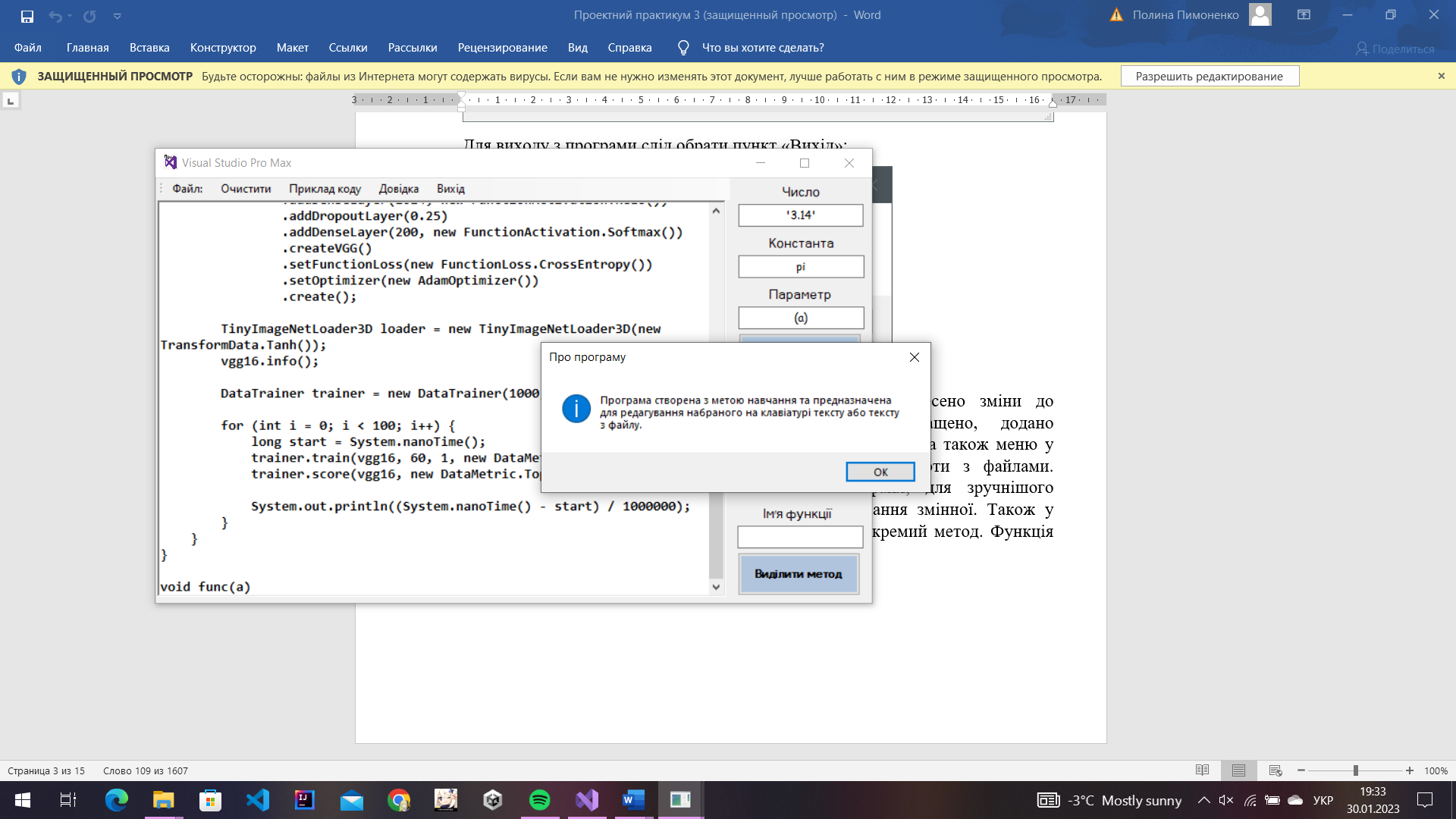
1. **Результати роботи програми**



4.1. число 3.14 у результатах тесту було замінено на символ pi



3.2. Додавання параметра



3.3. Довідка про програму

1. **Висновок**

У процесі виконання лабораторної роботи було зроблену «покращену» версію додатку з лабораторної роботи №2, додавши такі можливості: вертикальний, горізонтальний scroll, довідку, можливість відкриття/збереження файлу, більш зручний та детальний інтерфейс. У розділі «приклад коду» користувач може отримати код за шаблоном і також редагувати його