**ЛАБОРАТОРНА РОБОТА №4**

Рефакторинг програмного коду

**Виконав ст. гр. Кн-19-2 Селезньов О. О.**

**Мета роботи:** навчитися виконувати реорганізацію програмного коду на підставі шаблонів рефакторінгу.

**Хід виконання роботи**

1. Виконати аналіз програмного коду розроблюваного ПО і модульних тестів з метою виявлення погано організованого код.
2. Використовуючи шаблони рефакторінгу, виконати реорганізацію програмного коду розроблюваного ПО і модульних тестів.
3. Перевірити успішність виконання всіх модульних тестів.
4. Виконати опис вироблених операцій рефакторінгу (було-стало-шаблон рефакторінгу).

**Виконання завдання:**

1. **Конструктор класу ElementObject великий, тому вирішуємо проблеми через прийоми рефакторингу Pull Up Constructor Body та Extract Method.**

Конструктор буде розбитий на стандартний конструктор та додатковий. А також частина коду буде винесена в окрему функцію.

Було:

public ElementObject(Point startPos ,IElement elementData, string id = "", ElementParameter parameter = null)

{

Id = (id == "") ? Guid.NewGuid().ToString("N") : id;

this.elementData = elementData;

Parameters = (parameter != null) ? parameter : elementData.Parameters;

Parameters.Position = startPos;

Nodes = new List<Node>() {

new Node(NodePosition.Left, this),

new Node(NodePosition.Right, this),

new Node(NodePosition.Top, this),

new Node(NodePosition.Bottom, this)

};

Nodes.ForEach(item =>

{

item.Move(Parameters.Position, Parameters.CustomSize);

});

}

Стало:

public ElementObject(Point startPos, IElement elementData)

{

Id = Guid.NewGuid().ToString("N");

this.elementData = elementData;

Parameters = elementData.Parameters;

Parameters.Position = startPos;

InitNodes();

}

public ElementObject(Point startPos ,IElement elementData, string id, ElementParameter parameter) : this(startPos, elementData)

{

Id = id;

this.elementData = elementData;

Parameters = parameter;

InitNodes();

}

private void InitNodes()

{

Nodes = new List<Node>() {

new Node(NodePosition.Left, this),

new Node(NodePosition.Right, this),

new Node(NodePosition.Top, this),

new Node(NodePosition.Bottom, this)

};

Nodes.ForEach(item =>

{

item.Move(Parameters.Position, Parameters.CustomSize);

});

}

1. **Метод DrawElement виконує багато операцій, тому можна використовувати прийом рефакторингу Extract Method та розбити його на кілька методів.**

Метод DrawElement буде розбитий на 2 методи DrawText та DrawFrame.

Було:

Font drawFont = new Font("Microsoft Sans Serif", 14);

SolidBrush TextClr = new SolidBrush(Color.Black);

public void DrawElement(Graphics g, bool selected = false)

{

StringFormat sf = new StringFormat();

sf.LineAlignment = StringAlignment.Center;

sf.Alignment = StringAlignment.Center;

elementData.Draw(g, Parameters);

if(elementData.Name != "Text")

g.DrawString((Parameters.Text == "") ? elementData.Name : Parameters.Text, drawFont, TextClr,

new Rectangle(Parameters.Position, Parameters.CustomSize), sf);

if (selected)

using (Pen pen = new Pen(Color.Aqua, 3))

{

g.DrawRectangle(pen, new Rectangle(Parameters.Position, Parameters.CustomSize));

}

}

Стало:

public void DrawElement(Graphics g, bool selected = false)

{

elementData.Draw(g, Parameters);

if (elementData.Name != "Text")

DrawText(g);

if (selected)

DrawFrame(g);

}

private void DrawText(Graphics g)

{

using (StringFormat sf = new StringFormat())

using (SolidBrush TextClr = new SolidBrush(Color.Black))

using (Font drawFont = new Font("Microsoft Sans Serif", 14))

{

sf.LineAlignment = StringAlignment.Center;

sf.Alignment = StringAlignment.Center;

g.DrawString((Parameters.Text == "") ? elementData.Name : Parameters.Text, drawFont, TextClr,

new Rectangle(Parameters.Position, Parameters.CustomSize), sf);

}

}

private void DrawFrame(Graphics g)

{

using(Pen pen = new Pen(Color.Aqua, 3))

{

g.DrawRectangle(pen, new Rectangle(Parameters.Position, Parameters.CustomSize));

}

}

1. **У методі DeleteNode використовується незрозумілий код, його можна виправити за допомогою Extract Variable.**

Було:

public static void DeleteNode(Canvas canvas, Node node)

{

Line temp = null;

foreach (var item in canvas.Lines)

{

if(item.FirstNode == node || item.SecondNode == node)

{

temp = item;

}

}

canvas.Lines.Remove(temp);

}

Стало:

public static void DeleteNode(Canvas canvas, Node node)

{

var currentLines = canvas.Lines.FindAll(line => line.FirstNode == node || line.SecondNode == node);

currentLines.ForEach(line =>

{

canvas.Lines.Remove(line);

});

}

1. **Метод DeleteElement виконує багато операцій, тому можна використовувати прийом рефакторингу Extract Method та розбити його на кілька методів.**

Частина методу DeleteElement винесена в окремий метод DeleteNode.

Було:

public static void DeleteElement(this Canvas canvas, ElementObject element)

{

element.Nodes.ForEach(node =>

{

var temp = canvas.Lines.FindAll(line => line.FirstNode == node || line.SecondNode == node);

temp.ForEach(line =>

{

canvas.Lines.Remove(line);

});

});

canvas.Elements.Remove(element);

canvas.OnElementsChanged();

}

Стало:

public static void DeleteElement(this Canvas canvas, ElementObject element)

{

element.Nodes.ForEach(node =>

{

DeleteNode(canvas, node);

});

canvas.Elements.Remove(element);

canvas.OnElementsChanged();

}

public static void DeleteNode(Canvas canvas, Node node)

{

var temp = canvas.Lines.FindAll(line => line.FirstNode == node || line.SecondNode == node);

temp.ForEach(line =>

{

canvas.Lines.Remove(line);

});

}

1. **Методи CreateFile та Import виконують багато операцій, тому можна використовувати прийом рефакторингу Extract Method та розбити їх на кілька методів.**

Частини методу CreateFile винесені у 2 окремих методи ExportElement та ExportLine.

Частини методу Import винесені в 2 окремі методи LoadElements і LoadLines.

Було:

public void CreateFile(string path)

{

save = new SaveElement();

\_canvas.Elements.ForEach(element =>

{

save.elements.Add(new SaveElement.StructElement { Id = element.Id, parameter = element.Parameters,

elementData = element.elementData.Name});

});

\_canvas.Lines.ForEach(line =>

{

save.lines.Add(new SaveElement.StructLine { firstNodeId = line.FirstNode.Parent.Id,

firstNodePos = line.FirstNode.nodePosition.ToString(),

secondNodeId = line.SecondNode.Parent.Id,

secondNodePos = line.SecondNode.nodePosition.ToString() });

});

using(FileStream fs = new FileStream(path, FileMode.OpenOrCreate, FileAccess.ReadWrite))

{

\_formatter.Serialize(fs, save);

fs.Close();

}

}

public void Import(string path)

{

save = new SaveElement();

if (File.Exists(path))

{

\_canvas.ClearElements();

using (FileStream fs = new FileStream(path, FileMode.Open, FileAccess.Read))

{

save = (SaveElement)\_formatter.Deserialize(fs);

fs.Close();

save.elements.ForEach(item =>

{

\_canvas.AddElement(new ElementObject(item.parameter.Position, Form1.elements.Find(element => item.elementData.Contains(element.Name)), item.Id, item.parameter));

});

save.lines.ForEach(line =>

{

Node firstNode = FindNode(line.firstNodeId, line.firstNodePos);

Node secondNode = FindNode(line.secondNodeId, line.secondNodePos);

if (firstNode != null && secondNode != null)

\_canvas.Lines.Add(new Line(firstNode, secondNode));

});

}

}

}

Стало:

public void CreateFile(string path)

{

save = new SaveElement();

ExportElements();

ExportLines();

using(FileStream fs = new FileStream(path, FileMode.OpenOrCreate, FileAccess.ReadWrite))

{

\_formatter.Serialize(fs, save);

fs.Close();

}

}

private void ExportElements()

{

\_canvas.Elements.ForEach(element =>

{

save.elements.Add(new SaveElement.StructElement

{

Id = element.Id,

parameter = element.Parameters,

elementData = element.elementData.Name

});

});

}

private void ExportLines()

{

\_canvas.Lines.ForEach(line =>

{

save.lines.Add(new SaveElement.StructLine

{

firstNodeId = line.FirstNode.Parent.Id,

firstNodePos = line.FirstNode.nodePosition.ToString(),

secondNodeId = line.SecondNode.Parent.Id,

secondNodePos = line.SecondNode.nodePosition.ToString()

});

});

}

public void Import(string path)

{

save = new SaveElement();

if (File.Exists(path))

{

\_canvas.ClearElements();

using (FileStream fs = new FileStream(path, FileMode.Open, FileAccess.Read))

{

save = (SaveElement)\_formatter.Deserialize(fs);

fs.Close();

LoadElements();

LoadLines();

}

}

}

private void LoadElements()

{

save.elements.ForEach(item =>

{

\_canvas.AddElement(new ElementObject(item.parameter.Position, Form1.elements.Find(element => item.elementData.Contains(element.Name)), item.Id, item.parameter));

});

}

private void LoadLines()

{

save.lines.ForEach(line =>

{

Node firstNode = FindNode(line.firstNodeId, line.firstNodePos);

Node secondNode = FindNode(line.secondNodeId, line.secondNodePos);

if(firstNode != null && secondNode != null)

\_canvas.Lines.Add(new Line(firstNode, secondNode));

});

}

1. **Метод DrawLine великий і у ньому написані некоректні назви змінних.**

Потрібно зробити перейменування полів і частин методу DrawLine винести в 2 окремих методи CalculateMidpointAndDistance і CalculateNodePosition.

Було:

public void Draw(Graphics graphics)

{

using(AdjustableArrowCap bA = new AdjustableArrowCap(4, 4))

using (Pen p = new Pen(Color.Black, 5))

{

Point a = new Point(FirstNode.position.X + FirstNode.Size.Height / 2, FirstNode.position.Y + FirstNode.Size.Width / 2);

Point b = new Point(SecondNode.position.X + SecondNode.Size.Height / 2, SecondNode.position.Y + SecondNode.Size.Width / 2);

Point c = new Point(0, 0);

if (FirstNode.Parent.Parameters.PolyLine)

{

if ((FirstNode.nodePosition == NodePosition.Bottom || FirstNode.nodePosition == NodePosition.Top))

{

c.X = a.X;

c.Y = b.Y;

}

if ((FirstNode.nodePosition == NodePosition.Left || FirstNode.nodePosition == NodePosition.Right))

{

c.X = b.X;

c.Y = a.Y;

}

}

double distance = Math.Sqrt(Math.Pow(b.X - c.X, 2) + Math.Pow(b.Y - c.Y, 2));

List<Point> points = new List<Point>();

points.Add(a);

if (c.X != 0 && distance > 90)

{

points.Add(c);

}

points.Add(b);

p.CustomEndCap = bA;

graphics.DrawLines(p, points.ToArray());

}

}

Стало:

public void Draw(Graphics graphics)

{

using(AdjustableArrowCap bigArrow = new AdjustableArrowCap(4, 4))

using (Pen pen = new Pen(Color.Black, 5))

{

Point first = CalculateNodePosition(FirstNode);

Point second = CalculateNodePosition(SecondNode);

var (middle, distance) = CalculateMidpointAndDistance(first, second);

List<Point> points = new List<Point>();

points.Add(first);

if (middle.X != 0 && distance > 90)

{

points.Add(middle);

}

points.Add(second);

pen.CustomEndCap = bigArrow;

graphics.DrawLines(pen, points.ToArray());

}

}

private (Point, double) CalculateMidpointAndDistance(Point first, Point second)

{

Point middle = new Point(0, 0);

if (FirstNode.Parent.Parameters.PolyLine)

{

if ((FirstNode.nodePosition == NodePosition.Bottom || FirstNode.nodePosition == NodePosition.Top))

{

middle.X = first.X;

middle.Y = second.Y;

}

if ((FirstNode.nodePosition == NodePosition.Left || FirstNode.nodePosition == NodePosition.Right))

{

middle.X = second.X;

middle.Y = first.Y;

}

}

double distance = Math.Sqrt(Math.Pow(second.X - middle.X, 2) + Math.Pow(second.Y - middle.Y, 2));

return (middle, distance);

}

private Point CalculateNodePosition(Node node)

{

return new Point(node.position.X + node.Size.Height / 2, node.position.Y + node.Size.Width / 2);

}

1. **У методах pictureToolStripMenuItem\_Click та fileToolStripMenuItem\_Click відбувається дублювання коду (Duplicate Code). Тому використовуватимемо Extract Method.**

Було:

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

{

string filename = "";

saveFileDialog1.Filter = "Bitmap Image (.bmp)|\*.bmp|JPEG Image (.jpeg)|\*.jpeg|Png Image (.png)|\*.png|Tiff Image (.tiff)|\*.tiff|Wmf Image (.wmf)|\*.wmf";

if (saveFileDialog1.ShowDialog() != DialogResult.Cancel)

filename = saveFileDialog1.FileName;

if (filename != null)

try

{

pictureBox1.Image.Save(filename);

MessageBox.Show($"File created\n{filename}");

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

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

{

string filename = "";

saveFileDialog1.Filter = "blockSheme (.block)|\*.block";

if (saveFileDialog1.ShowDialog() != DialogResult.Cancel)

filename = saveFileDialog1.FileName;

if (filename != null)

try

{

\_fileSystem.CreateFile(filename);

MessageBox.Show($"File created\n{filename}");

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

Стало:

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

{

SaveDialog("Bitmap Image (.bmp)|\*.bmp|JPEG Image (.jpeg)|\*.jpeg|Png Image (.png)|\*.png|Tiff Image (.tiff)|\*.tiff|Wmf Image (.wmf)|\*.wmf", (fileName) =>

{

\_fileSystem.CreateFile(fileName);

MessageBox.Show($"File created\n{fileName}");

});

}

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

{

SaveDialog("blockSheme (.block)|\*.block", (fileName) =>

{

\_fileSystem.CreateFile(fileName);

MessageBox.Show($"File created\n{fileName}");

});

}

private void SaveDialog(string filter, Action<string> callBack)

{

saveFileDialog1.Filter = filter;

if (saveFileDialog1.ShowDialog() != DialogResult.Cancel)

{

try

{

callBack(saveFileDialog1.FileName);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

}

1. **У класі ElementParameter у полів є код, що дублюється, його можна винести в окремий метод.**

Було:

private double \_angle = int.MinValue;

public double Angle

{

get { if (\_angle >= 110) return 110; if (\_angle <= 0 && \_angle != int.MinValue) return 1; return \_angle; }

set { \_angle = value; }

}

private double \_size = int.MinValue;

public double FontSize

{

get { if (\_size >= 90) return 90; if (\_size < 4 && \_size != int.MinValue) return 4; return \_size; }

set { \_size = value; }

}

private int \_verticesCount = int.MinValue;

public int VerticesCount { get { if (\_verticesCount >= 9) return 9; if (\_verticesCount < 4 && \_verticesCount != int.MinValue) return 4; return \_verticesCount; }

set { \_verticesCount = value; } }

Стало:

private double \_angle = int.MinValue;

public double Angle

{

get { return MinMax(1, 110, \_angle); }

set { \_angle = value; }

}

private double \_size = int.MinValue;

public double FontSize

{

get { return MinMax(4, 90, \_size); }

set { \_size = value; }

}

private int \_verticesCount = int.MinValue;

public int VerticesCount { get { return (int)MinMax(4, 9, \_verticesCount); }

set { \_verticesCount = value; } }

private double MinMax(int min, int max, double baseValue)

{

if (baseValue >= max)

return max;

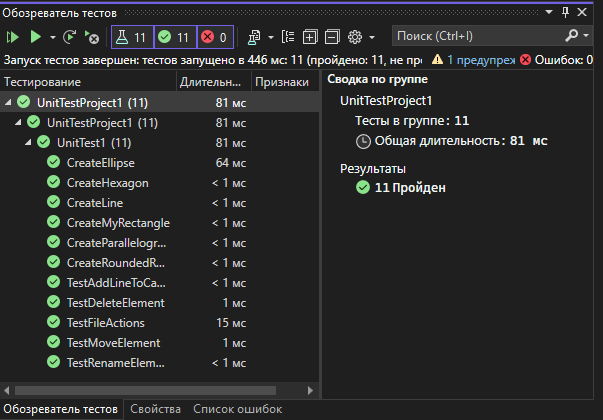
if (baseValue < min && baseValue != int.MinValue)

return min;

return baseValue;

}

Перевіримо успішність виконання всіх модульних тестів, після рефакторингу:



**Висновок:** навчився виконувати реорганізацію програмного коду на підставі шаблонів рефакторінгу.