УО «Белорусский государственный университет информатики и радиоэлектроники»

Кафедра ПОИТ

Отчет по лабораторной работе № 2

по предмету «Объектно-Ориентированные Технологии Программирования и Стандарты Проектирования»

Выполнил:

Шиманчук А.О.

гр. 351001

Проверил:

Деменковец Д.В.

Минск 2025

public partial class Form1 : Form

{

private Figure CurrentFigure = null;

private FigureList ListOfFigures = new FigureList();

public Form1()

{

InitializeComponent();

this.Width = 1000;

this.Height = 800;

this.FormBorderStyle = FormBorderStyle.FixedSingle;

this.MaximizeBox = false;

this.StartPosition = FormStartPosition.CenterScreen;

}

private void Form1\_MouseClick(object sender, MouseEventArgs e)

{

if (e.Y <= 200 || FigurecomboBox.SelectedIndex == -1 || textBox1.Text == "" || textBox2.Text == "")

return;

string StrFigure = FigurecomboBox.SelectedItem.ToString();

int param1 = Convert.ToInt32(textBox1.Text);

int param2 = Convert.ToInt32(textBox2.Text);

if (param2 <= 200 && StrFigure == "Линия")

return;

CurrentFigure = Factory.CreateShape(StrFigure, e.X, e.Y, param1, param2);

ListOfFigures.AddList(CurrentFigure, HistoryListBox);

this.Invalidate();

}

private void Form1\_Paint(object sender, PaintEventArgs e)

{

Graphics g = CreateGraphics();

g.DrawLine(Pens.Black, 0, 200, this.Width, 200);

g.FillRectangle(Brushes.AntiqueWhite, 0, 0, this.Width, 200);

ListOfFigures.DrawFromList(g);

}

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

{

if (HistoryListBox.Items.Count > 0)

{

ListOfFigures.RemoveLastChild(HistoryListBox);

this.Invalidate();

}

}

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

{

ListOfFigures.RemoveList(HistoryListBox);

this.Invalidate();

}

private void FigurecomboBox\_SelectedIndexChanged(object sender, EventArgs e)

{

switch (FigurecomboBox.SelectedIndex)

{

case 0:

{

textBox2.Visible = true;

label2.Visible = true;

label1.Text = "Ширина";

label2.Text = "Высота";

break;

}

case 1:

{

textBox2.Visible = true;

label2.Visible = true;

label1.Text = "Ширина";

label2.Text = "Высота";

break;

}

case 2:

{

label1.Text = "Ширина и высота";

label2.Visible = false;

textBox2.Visible = false;

break;

}

case 3:

{

label2.Visible = true;

textBox2.Visible = true;

label1.Text = "Ширина";

label2.Text = "Высота";

break;

}

case 4:

{

label1.Text = "Радиус";

label2.Visible = false;

textBox2.Visible = false;

break;

}

case 5:

{

label2.Visible = true;

textBox2.Visible = true;

label1.Text = "Координата x";

label2.Text = "Координата y";

break;

}

}

}

}

internal static class Factory

{

private static Dictionary<string, Func<int, int, int, int, Figure>> \_shapeCreators = new Dictionary<string, Func<int, int, int, int, Figure>>();

static Factory()

{

RegisterShape("Эллипс", (x, y, width, height) => new Ellipse(x, y, width, height));

RegisterShape("Круг", (x, y, width, height) => new Circle(x, y, width, width));

RegisterShape("Прямоугольник", (x, y, width, height) => new Rectangle(x, y, width, height));

RegisterShape("Линия", (x, y, x1, y1) => new Line (x, y, x1, y1));

RegisterShape("Параллелограмм", (x, y, width, height) => new Parallelogram(x, y, width, height));

RegisterShape("Квадрат", (x, y, width, height) => new Square(x, y, width, width));

}

public static void RegisterShape(string shapeType, Func<int, int, int, int, Figure> creator)

{

\_shapeCreators[shapeType] = creator;

}

public static Figure CreateShape(string shapeType, int x, int y, int p1, int p2)

{

if (\_shapeCreators.ContainsKey(shapeType))

{

return \_shapeCreators[shapeType](x, y, p1, p2);

}

return null;

}

}

public abstract class Figure

{

protected int X { get; set; }

protected int Y { get; set; }

public Figure(int x, int y)

{

X = x;

Y = y;

}

public Figure() { }

public abstract void Draw(Graphics g);

public abstract void AddToListBox(ListBox listBox);

}

public class Parallelogram : Figure

{

protected int Width { get; set; }

protected int Height { get; set; }

public Parallelogram(int x, int y,int width, int height) : base(x, y)

{

Width = width;

Height = height;

}

public override void Draw(Graphics g)

{

int offset = 20;

Point[] PointsForPolygon = new Point[]

{

new Point(X, Y),

new Point(X + Width, Y),

new Point(X + Width - offset, Y + Height),

new Point(X - offset, Y + Height)

};

g.DrawPolygon(Pens.Black, PointsForPolygon);

g.FillPolygon(Brushes.Aquamarine, PointsForPolygon);

}

public override void AddToListBox(ListBox listBox)

{

listBox.Items.Add("Паралелограмм");

}

}

public class FigureList

{

private List<Figure> ListOfFigures = new List<Figure>();

public void AddList(Figure figure, ListBox list)

{

ListOfFigures.Add(figure);

figure.AddToListBox(list);

}

public void RemoveList(ListBox list)

{

ListOfFigures.Clear();

list.Items.Clear();

}

public void RemoveLastChild(ListBox list)

{

ListOfFigures.RemoveAt(ListOfFigures.Count - 1);

list.Items.RemoveAt(list.Items.Count - 1);

}

public void DrawFromList(Graphics g)

{

foreach (Figure f in ListOfFigures)

{

f.Draw(g);

}

}

}