Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

Лабораторна робота №3

з дисципліни
«ООП»

Виконав:
студент групи ІМ-44
Куцевіл Іван Павлович
номер у списку групи: 12

Перевірив:
Порєв ВікторМиколайович.

Київ 2025

Завдання:

Ствоити програму, завдяки якій можна додавати прості геометричні фігури у вікно, використовуючи принципи поліморфізму. Фігури мають обиратися через запрограмований інтерфейс користувача. Властивості фігур і “гумового” сліду визначаються за варіантом.

Отже, згідно мого варіанту: гумовий слід – суцільна лінія червоного кольору; масив вказівників для фігур – статичний; прямокутник – вводиться з центру до одного з протилежних кутів, заповнення – немає, лише чорний контур; еліпс – вводиться по двом протилежним кутам охоплюючого прямокутника, заповнення – жовтий колір з чорним контуром. Позначка поточного типу об’єкту, що вводиться: в заголовку вікна.

Код програми:

Shape.cs:

```
using System;
using System.Drawing;

namespace OOP_Lab3
{
    public abstract class Shape
    {
        public abstract void PreDraw(ShapePosition shapePosition, Graphics graphics);
        public abstract void Draw(ShapePosition shapePosition, Graphics graphics);
    }

    public class Dot : Shape
    {
        public override void PreDraw(ShapePosition shapePosition, Graphics graphics)
        {
            return;
        }

        public override void Draw(ShapePosition shapePosition, Graphics graphics)
        {
            graphics.FillEllipse(Brushes.Red, shapePosition.startPostion.X - 2,
shapePosition.startPostion.Y - 2, 5, 5);
        }
    }

    public class Line : Shape
    {
        public override void PreDraw(ShapePosition shapePosition, Graphics graphics)
        {
            graphics.DrawLine(Pens.Red, shapePosition.startPostion.X, shapePosition.startPostion.Y,
shapePosition.lastPostion.X, shapePosition.lastPostion.Y);
        }

        public override void Draw(ShapePosition shapePosition, Graphics graphics)
        {
            graphics.DrawLine(Pens.Black, shapePosition.startPostion.X, shapePosition.startPostion.Y,
shapePosition.lastPostion.X, shapePosition.lastPostion.Y);
        }
    }

    public class Rectangle : Shape
    {
        public override void PreDraw(ShapePosition shapePosition, Graphics graphics)
        {
            graphics.DrawRectangle(Pens.Red, GetRectangle(shapePosition));
        }

        public override void Draw(ShapePosition shapePosition, Graphics graphics)
        {
            graphics.DrawRectangle(Pens.Black, GetRectangle(shapePosition));
        }
        private System.Drawing.Rectangle GetRectangle(ShapePosition shapePosition)
        {
            int centerX = shapePosition.startPostion.X;
            int centerY = shapePosition.startPostion.Y;

            int radiusX = Math.Abs(shapePosition.lastPostion.X - centerX);
            int radiusY = Math.Abs(shapePosition.lastPostion.Y - centerY);

            return new System.Drawing.Rectangle(centerX - radiusX, centerY - radiusY, radiusX * 2, radiusY
* 2);
        }
    }
```

```
        public class Circle : Shape
        {
        public override void PreDraw(ShapePosition shapePosition, Graphics graphics)
        {
            graphics.DrawEllipse(Pens.Red, GetRectangle(shapePosition));
        }

        public override void Draw(ShapePosition shapePosition, Graphics graphics)
        {
            System.Drawing.Rectangle rect = GetRectangle(shapePosition);
            graphics.FillEllipse(new SolidBrush(Color.Yellow), rect);
            graphics.DrawEllipse(Pens.Black, rect);
        }

        private System.Drawing.Rectangle GetRectangle(ShapePosition shapePosition)
        {
            int x = Math.Min(shapePosition.startPostion.X, shapePosition.lastPostion.X);
            int y = Math.Min(shapePosition.startPostion.Y, shapePosition.lastPostion.Y);
            int width = Math.Abs(shapePosition.startPostion.X - shapePosition.lastPostion.X);
            int height = Math.Abs(shapePosition.startPostion.Y - shapePosition.lastPostion.Y);

            return new System.Drawing.Rectangle(x, y, width, height);
        }
    }
}
```

ShapePosition.cs:

```
using System.Drawing;

namespace OOP_Lab3
{
    public class ShapePosition
    {
        public Point startPostion;
        public Point lastPostion;
    }
}
```

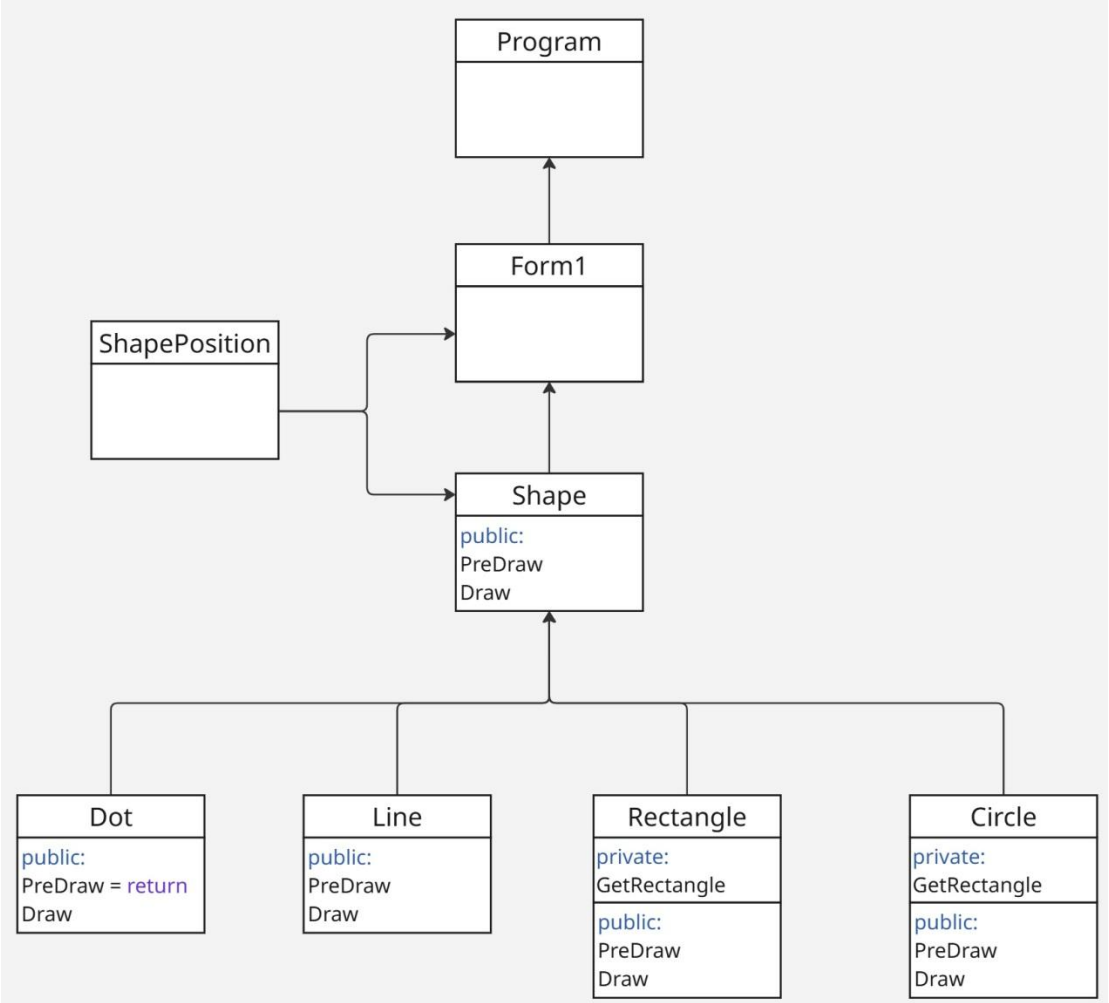
Form1.cs:

```
using System.Collections.Generic;
using System.Drawing;
using System.Windows.Forms;

namespace OOP_Lab3
{
    public partial class Form1 : Form
    {
        private bool isMouseButtonDown = false;

        private ShapePosition shapePosition = new ShapePosition();
        private Shape currentShape = null;
        private Shape[] shapes;
        private int maxShapes = 113;

        private Bitmap bitmap;

        private Dictionary<string, Shape> shapeDictionary = new Dictionary<string, Shape>();

        public Form1()
        {
            InitializeComponent();

            bitmap = new Bitmap(pictureBox1.Width, pictureBox1.Height);
            shapes = new Shape[maxShapes];
            pictureBox1.Image = bitmap;
            shapePosition.startPostion = Point.Empty;
```

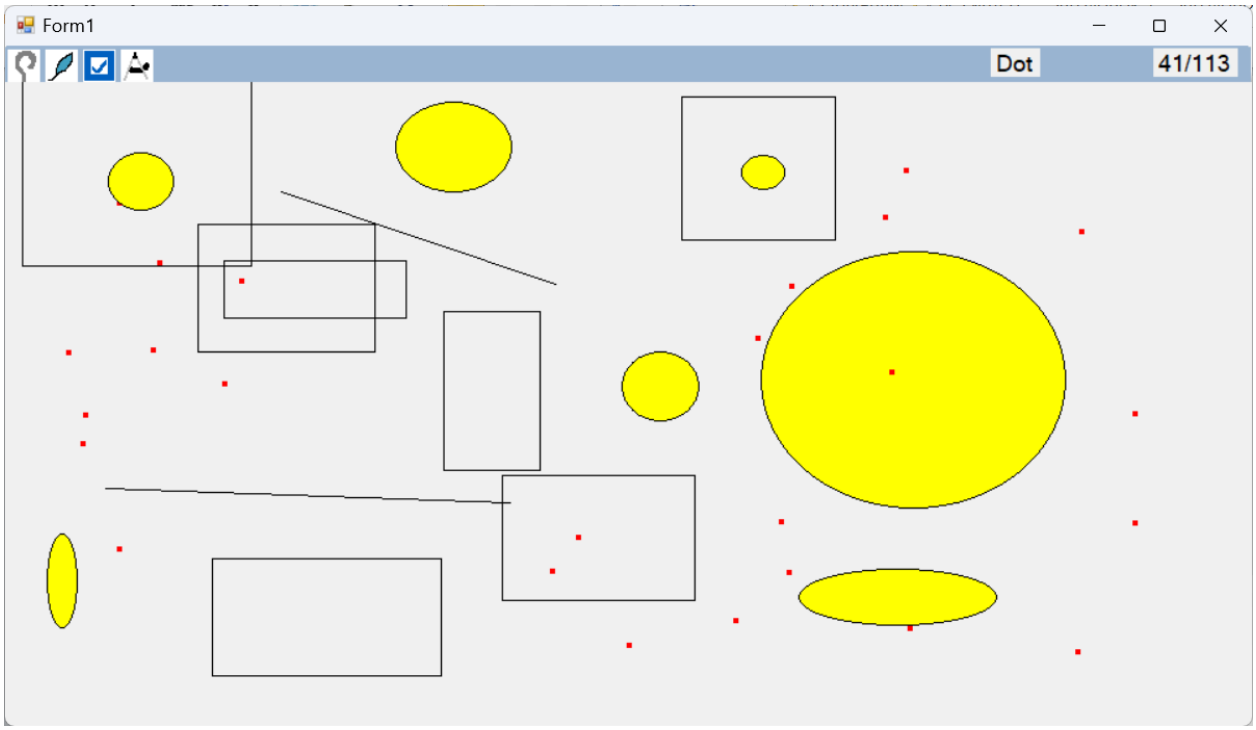
```
    shapeDictionary.Add("Dot", new Dot());
    shapeDictionary.Add("Line", new Line());
    shapeDictionary.Add("Rectangle", new Rectangle());
    shapeDictionary.Add("Circle", new Circle());

    UpdateShapesCountText();
}

private void toolStrip1_ItemClicked(object sender, ToolStripItemClickedEventArgs e)
{
    string shapeName = e.ClickedItem.Name.Split('_')[0];
    shapeDictionary.TryGetValue(shapeName, out currentShape);
    UpdateSelectedShapeText(shapeName);
}

private void pictureBox1_Paint(object sender, PaintEventArgs e)
{
    e.Graphics.DrawImage(bitmap, Point.Empty);
    if (isMouseButtonDown) currentShape.PreDraw(shapePosition, e.Graphics);
}

private void pictureBox1_MouseDown(object sender, MouseEventArgs e)
{
    if (GetShapesCount() < maxShapes && currentShape != null)
    {
        isMouseButtonDown = true;
        shapePosition.startPostion = ReadMousePosition();
    }
}

private void pictureBox1_MouseMove(object sender, MouseEventArgs e)
{
    if (isMouseButtonDown && GetShapesCount() < maxShapes && currentShape != null)
    {
        shapePosition.lastPostion = ReadMousePosition();
        pictureBox1.Invalidate();
    }
}

private void pictureBox1_MouseUp(object sender, MouseEventArgs e)
{
    if (GetShapesCount() < maxShapes && currentShape != null)
    {
        isMouseButtonDown = false;
        shapePosition.lastPostion = ReadMousePosition();
        Graphics graphics = Graphics.FromImage(bitmap);

        currentShape.Draw(shapePosition, graphics);
        AddShape(currentShape);
        pictureBox1.Invalidate();
    }
}

private Point ReadMousePosition()
{
    Point screenPos = Cursor.Position;
    return pictureBox1.PointToClient(screenPos);
}

private void AddShape(Shape shape)
{
    for (int i = 0; i < shapes.Length; i++)
    {
        if (shapes[i] == null)
        {
            shapes[i] = shape;
            break;
        }
    }

    UpdateShapesCountText();
}
```

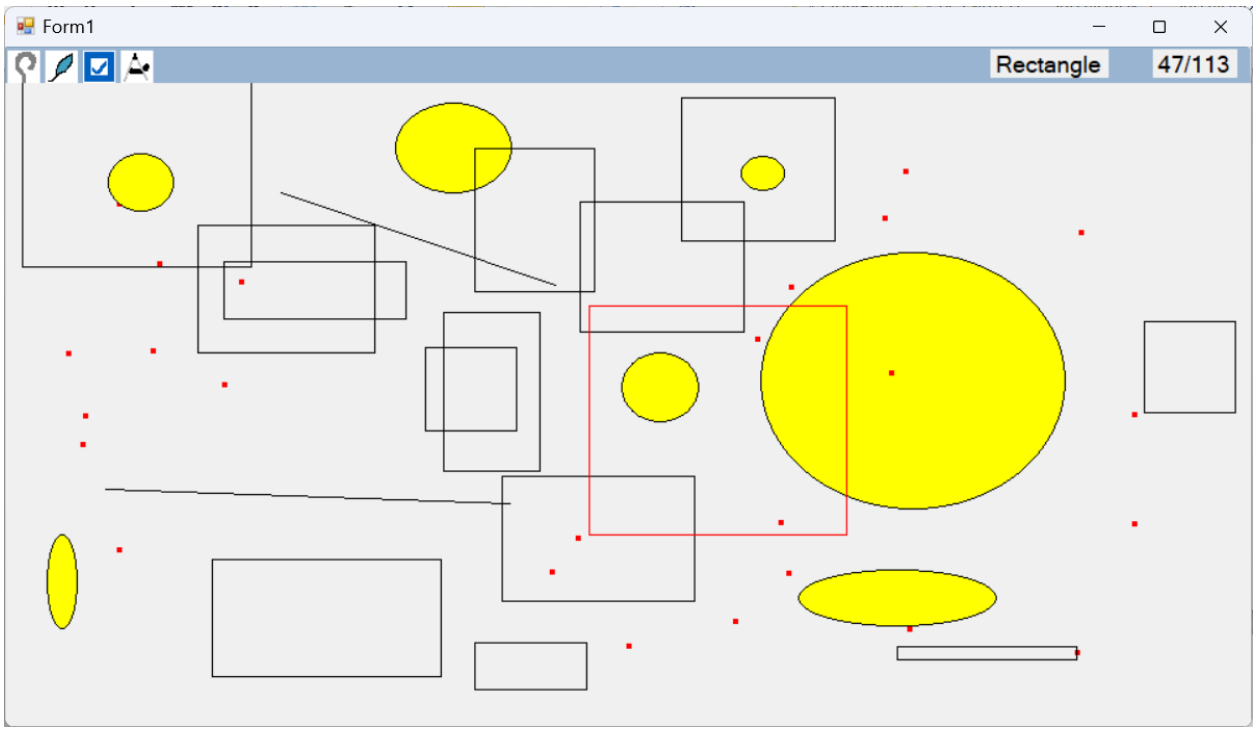
```
        private int GetShapesCount()
        {
            int shapesCount = 0;

            foreach (Shape shapeInList in shapes)
            {
                if (shapeInList != null)
                {
                    shapesCount++;
                }
                else
                {
                    break;
                }
            }

            return shapesCount;
        }

        private void UpdateShapesCountText()
        {
            label1.Text = GetShapesCount() + "/" + maxShapes;
        }

        private void UpdateSelectedShapeText(string shapeName)
        {
            label2.Text = shapeName;
        }
    }
}
```

Діаграми класів:



Результати виконання:





Висновок:

Під час роботи, я створив програму, завдяки якій можна додавати прості геометричні фігури у вікно, використовуючи інтерфейс користувача. Отримав вміння та навички використовувати інкапсуляцію, абстракцію типів, успадкування та поліморфізм на основі класів, запрограмувавши простий графічний редактор в об’єктно-орієнтованому стилі.