**Лабораторна робота №9**

*Проектування класів та компонент*

**Мета:** Набути навиків створення нащадків класів. Засвоєння принципу ( наслідування( спадковості) в ООП(C#).

**Постановка задачі:** Розробити програму, після запуску якої на формі з’являється анімоване зображення многокутника: вершини рухаються під кутом в різних напрямках і відбиваються від меж форми.

### Завдання для вдосконалення проекту

1. Доповнити компонент Katok встановленням випадкових кольорів фону та ліній, причому кольори мають відрізнятися так, щоб лінії було добре видно.
2. Забезпечити відображення одночасно п’ятьох послідовних фаз руху. При цьому малюється 1-ше, 2-е, 3-є, 4-е, 5-те положення многокутника. Потім стирається 1-е і домальовується 6-те, стирається 2-е і домальовується 7-е і т.п.

### Завдання для самостійної розробки

1. Створити проект, у якому було б одночасно декілька рухомих многокутників:
   1. в різних частинах екрану (декілька компонентів Katok);
   2. в одному прямокутнику (змінений компонент Katok).

**Програма:**

**KatokForm.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Lab\_9\_Ex\_Katok

{

public partial class KatokForm : Form

{

Katok[] katok;

public KatokForm()

{

InitializeComponent();

Random random = new Random();

Form form = Application.OpenForms["MainForm"];

katok = new Katok[Convert.ToInt32(((MainForm)form).numericUpDown2.Value)];

for (int i = 0; i < Convert.ToInt32(((MainForm)form).numericUpDown2.Value); i++)

{

katok[i] = new Katok(Convert.ToInt32(((MainForm)form).numericUpDown1.Value));

katok[i].Parent = this;

katok[i].Size = ClientSize;

katok[i].Anchor = (AnchorStyles.Bottom | AnchorStyles.Left | AnchorStyles.Right | AnchorStyles.Top);

}

}

}

}

**Katok.cs**

using System;

using System.Collections.Generic;

using System.Drawing;

using System.IO;

using System.Windows.Forms;

using System.Text.Json;

namespace Lab\_9\_Ex\_Katok

{

public partial class Katok : UserControl

{

public List<color> colors = new List<color>();

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

{

using (FileStream fs = new FileStream("colors.json", FileMode.OpenOrCreate))

{

colors = JsonSerializer.Deserialize<List<color>>(fs);

}

}

public Point[] point;

private int[] dx, dy;

private Random rand = new Random();

private int n;

public Katok(int N)

{

InitializeComponent();

n = N;

point = new Point[n];

dx = dy = new int[n];

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

{

point[i].X = rand.Next(Width);

point[i].Y = rand.Next(Height);

do { dx[i] = rand.Next(6) - 3; } while (dx[i] == 0);

do { dy[i] = rand.Next(6) - 3; } while (dy[i] == 0);

}

timer1.Start();

}

private void timer1\_Tick(object sender, EventArgs e)

{

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

{

point[i].X += dx[i];

point[i].Y += dy[i];

if ((point[i].X < 3) || (point[i].X > Width - 3))

dx[i] = -dx[i];

if ((point[i].Y < 3) || (point[i].Y > Height - 3))

dy[i] = -dy[i];

if (point[i].X > Width - 3) point[i].X = Width - 3;

if (point[i].Y > Height - 3) point[i].Y = Height - 3;

}

Invalidate();

}

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

{

Random rand = new Random();

color lineColor = colors[rand.Next(0, 99)];

Pen pen = new Pen(lineColor.getColor());

pen.Width += 3;

e.Graphics.DrawPolygon(pen, point);

color bgColor;

bgColor = colors[rand.Next(0,99)];

if (lineColor - bgColor > 100)

{

this.BackColor = bgColor.getColor();

}

}

}

}

**HomeForm.cs**

using System;

using System.Collections.Generic;

using System.Windows.Forms;

using System.Text.Json;

using System.IO;

namespace Lab\_9\_Ex\_Katok

{

public partial class MainForm : Form

{

public MainForm()

{

InitializeComponent();

}

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

{

KatokForm katokForm = new KatokForm();

katokForm.ShowDialog();

}

public List<color> colors = new List<color>();

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

{

using (FileStream fs = new FileStream("colors.json", FileMode.OpenOrCreate))

{

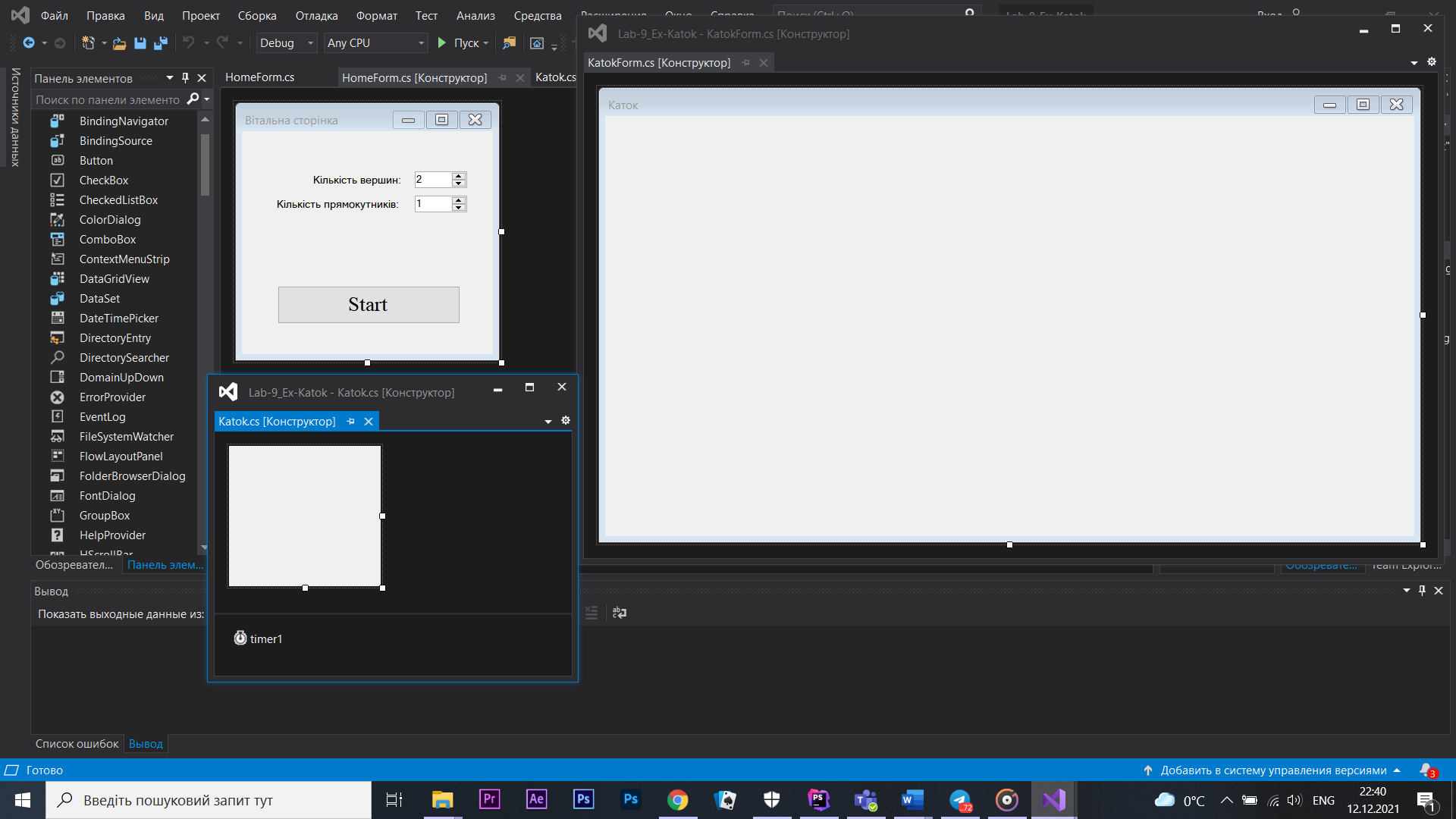
colors = JsonSerializer.Deserialize<List<color>>(fs);

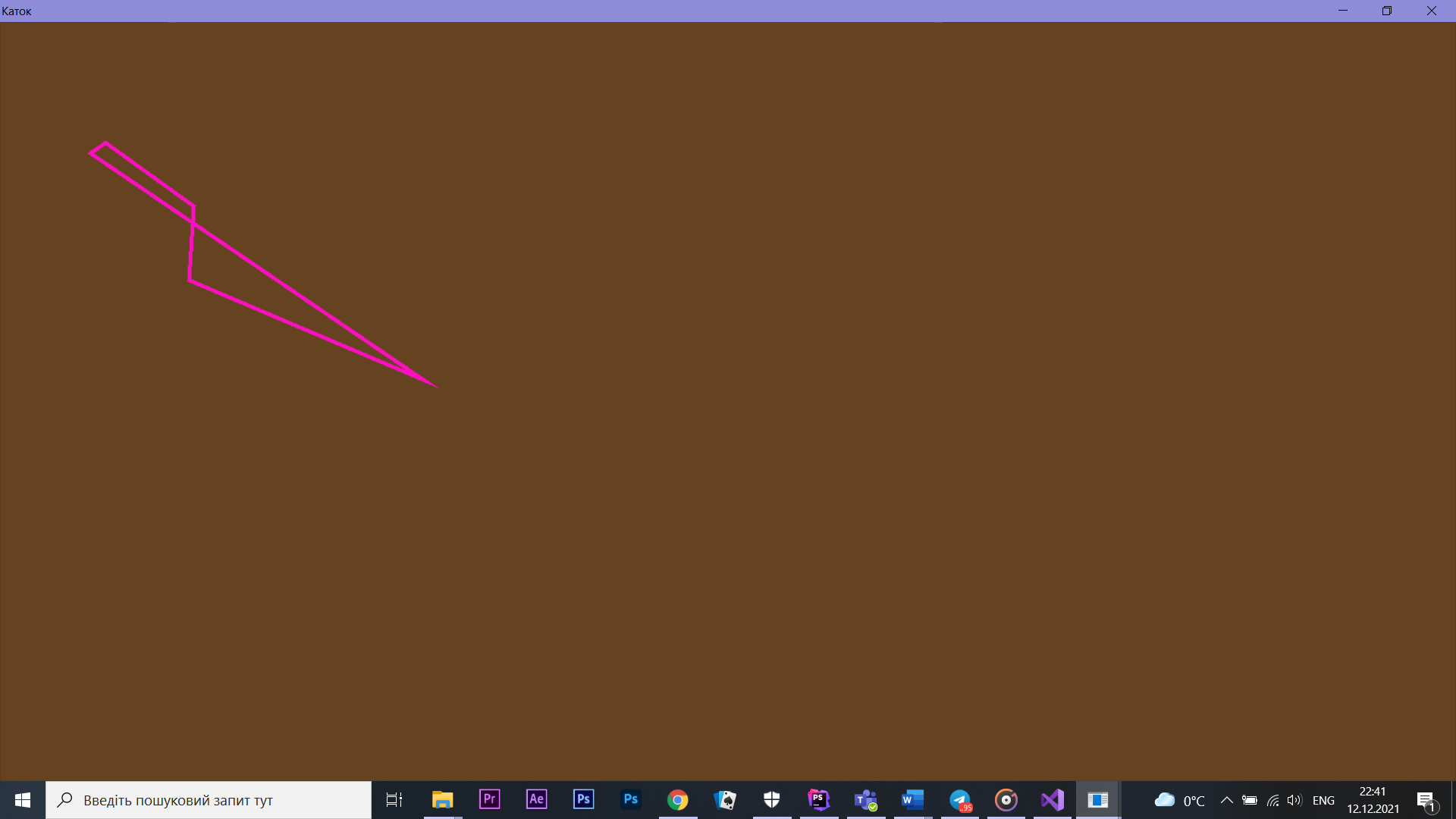
}

}

}

}

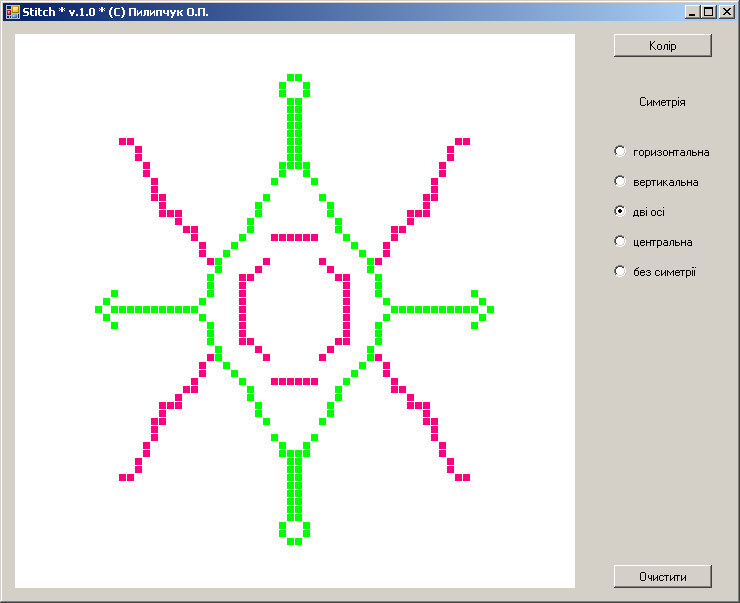




**Постановка задачі:** Розробити програму - редактор візерунків для вишивки хрестиком.

### Детальний опис функціонування проекту

1. Зовнішній вигляд вікна програми під час роботи показаний на малюнку:



1. Кнопкою «Колір» відкривається стандартне діалогове вікно для вибору кольору.
2. При клацанні в межах білого прямокутника на ньому з’являються квадратики, з яких складається візерунок.
3. Залежно від обраного виду симетрії (група радіокнопок), з’являються також квадратики, розміщені симетрично відносно певних осей (вертикальної, горизонтальної, обох) або центра малюнка.
4. Кнопкою «Очистити» візерунок стирається, а програма залишається готовою для подальшої роботи.

### План розробки

1. Розробити компонент Polotno – спадкоємець UserControl з такими властивостями:
   1. при клацанні його поверхні зберігається інформація про координати майбутнього квадратика і він з’являється на екрані. Також, за потреби, з’являються симетричні квадратики;
   2. метод Clear призводить до знищення інформації про наявний візерунок і очищення малюнка;
2. Встановити на форму компоненти у відповідності до малюнка.
3. Налаштувати дію радіокнопок.

### Завдання для вдосконалення проекту

1. Забезпечити збереження візерунка на диск та завантаження його з диска для продовження роботи.
2. Забезпечити збереження візерунка на диск у графічному форматі.
3. Забезпечити друкування візерунка на папері.
4. Забезпечити побудову суцільних ліній перетягуванням мишею.
5. Забезпечити «заливання» суцільної області обраним кольором.

**Polotno.cs**

using System;

using System.Drawing;

using System.Windows.Forms;

namespace Lab\_9\_Redaktor

{

public partial class Polotno : UserControl

{

public Polotno()

{

InitializeComponent();

}

public Color ActiveColor = Color.Red;

public string protokol = string.Empty;

private bool doPaint = false;

public enum Symetry // вид симетрії

{

NoAxis, // немає осі

VerticalAxis, // вертикальна вісь

HorizontalAxis, // горизонтальна вісь

TwoAxis, // дві осі

CentralAxis // центральна вісь

}

public Symetry symetryNumb = Symetry.NoAxis;

public int symm = 0;

public int CWidth, CHeight;

private int CrossWidth = 8;

private void AddCross(MouseEventArgs e, int x, int y)

{

int X = x/\* / CrossWidth \* CrossWidth\*/;

int Y = y/\* / CrossWidth \* CrossWidth\*/;

for (int p = 0; p < protokol.Length; p += 26)

if ((X == int.Parse(protokol.Substring(p, 5))) && Y == int.Parse(protokol.Substring(p + 5, 5)))

{ protokol = protokol.Remove(p, 26); break; }

protokol += string.Format("{0,5}{1,5}{2,4}{3,4}{4,4}{5,4}", X, Y, ActiveColor.A.ToString(), ActiveColor.R.ToString(), ActiveColor.G.ToString(), ActiveColor.B.ToString());

}

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

{

SolidBrush brush = new SolidBrush(ActiveColor);

for (int x = 0; x < protokol.Length; x += 26)

{// малюємо прямокутнички за даними протоколу

brush.Color = Color.FromArgb(int.Parse(protokol.Substring(x + 10, 4)), int.Parse(protokol.Substring(x + 14, 4)), int.Parse(protokol.Substring(x + 18, 4)), int.Parse(protokol.Substring(x + 22, 4)));

e.Graphics.FillRectangle(brush, int.Parse(protokol.Substring(x, 5)), int.Parse(protokol.Substring(x + 5, 5)), CrossWidth - 1, CrossWidth - 1);

}

}

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

{

AddCross(e, e.X, e.Y);

if (symetryNumb == Symetry.TwoAxis || symetryNumb == Symetry.CentralAxis)

AddCross(e, CWidth \* (CrossWidth) - e.X - 1, CHeight \* CrossWidth - e.Y - 1);

if ((symetryNumb == Symetry.VerticalAxis) || (symetryNumb == Symetry.TwoAxis))

AddCross(e, CWidth \* (CrossWidth) - e.X - 1, e.Y);

if ((symetryNumb == Symetry.HorizontalAxis) || (symetryNumb == Symetry.TwoAxis))

AddCross(e, e.X, CHeight \* (CrossWidth) - e.Y - 1);

Invalidate();

}

private void Polotno\_SizeChanged(object sender, EventArgs e)

{

CWidth = Width / CrossWidth;

CHeight = Height / CrossWidth;

}

public void Clear()

{

protokol = string.Empty;

Invalidate();

}

private void Polotno\_MouseDown(object sender, MouseEventArgs e)

{

doPaint = true;

}

private void Polotno\_MouseUp(object sender, MouseEventArgs e)

{

doPaint = false;

}

private void Polotno\_MouseLeave(object sender, EventArgs e)

{

doPaint = false;

}

private void Polotno\_MouseMove(object sender, MouseEventArgs e)

{

if (doPaint)

{

Polotno\_MouseClick(sender, e);

}

}

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

{

}

}

}

**Picture\_Editor.cs**

using System;

using System.Drawing;

using System.IO;

using System.Windows.Forms;

namespace Lab\_9\_Redaktor

{

public partial class PictureEditor : Form

{

public PictureEditor()

{

InitializeComponent();

openFileDialog1.Filter = saveFileDialog1.Filter = "Binary files(\*.dat)|\*.dat";

openFileDialog1.FileName = string.Empty;

saveFileDialog1.FileName = savePictureDialog1.FileName = "myPainting";

savePictureDialog1.Filter =

"PNG|\*.png|" +

"JPEG|\*.jpeg; \*.jpg; \*.jpe; \*.jfif|" +

"BMP|\*.bmp|" +

"GIF pictures|\*.gif";

}

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

{

colorDialog1.ShowDialog();

polotno1.ActiveColor = colorDialog1.Color;

}

private void Clear\_Button\_Click(object sender, EventArgs e)

{

polotno1.Clear();

}

private void RB\_Horizontal\_CheckedChanged(object sender, EventArgs e)

{

polotno1.symetryNumb = Polotno.Symetry.HorizontalAxis;

}

private void RB\_Vertical\_CheckedChanged(object sender, EventArgs e)

{

polotno1.symetryNumb = Polotno.Symetry.VerticalAxis;

}

private void RB\_Double\_CheckedChanged(object sender, EventArgs e)

{

polotno1.symetryNumb = Polotno.Symetry.TwoAxis;

}

private void RB\_Central\_CheckedChanged(object sender, EventArgs e)

{

polotno1.symetryNumb = Polotno.Symetry.CentralAxis;

}

private void RB\_NoSym\_CheckedChanged(object sender, EventArgs e)

{

polotno1.symetryNumb = Polotno.Symetry.NoAxis;

}

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

{

}

private void writeToFile(string protokol, string path)

{

using (BinaryWriter writer = new BinaryWriter(File.Open(path, FileMode.Create)))

{

writer.Write(protokol);

}

}

private void файлToolStripMenuItem1\_Click(object sender, EventArgs e)

{

if (saveFileDialog1.ShowDialog() == DialogResult.Cancel) return;

string path = saveFileDialog1.FileName;

writeToFile(polotno1.protokol, path);

}

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

{

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

string path = openFileDialog1.FileName;

using (BinaryReader reader = new BinaryReader(File.Open(path, FileMode.Open)))

{

polotno1.protokol = reader.ReadString();

polotno1.Invalidate();

}

}

private Bitmap SavePicture()

{

int k = 30;

Bitmap picture = new Bitmap(polotno1.Width + k, polotno1.Height + k);

for (int x = 0; x < picture.Width; x++)

{

for (int y = 0; y < picture.Height; y++)

{

picture.SetPixel(x, y, Color.White);

}

}

var temp = polotno1.protokol.Trim();

temp = System.Text.RegularExpressions.Regex.Replace(temp, @"\s+", " ");

string tempNumb = string.Empty;

int[] Point = new int[6];

for (int i = 0, ArrayPlace = 0; i < temp.Length; i++)

{

if (temp[i] == ' ')

{

Point[ArrayPlace] = Convert.ToInt32(tempNumb);

//MessageBox.Show(tempNumb);

tempNumb = string.Empty;

ArrayPlace++;

}

else

{

if (ArrayPlace == 6)

{

ArrayPlace = 0;

picture.SetPixel(Point[0], Point[1],

Color.FromArgb(Point[2], Point[3], Point[4], Point[5]));

}

tempNumb += temp[i];

}

}

return picture;

}

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

{

if (savePictureDialog1.ShowDialog() == DialogResult.Cancel) return;

SavePicture().Save(savePictureDialog1.FileName);

}

private void друкToolStripMenuItem1\_Click(object sender, EventArgs e)

{

printPreviewDialog1.ShowDialog();

printDocument1.Print();

}

private void printDocument1\_PrintPage(object sender, System.Drawing.Printing.PrintPageEventArgs e)

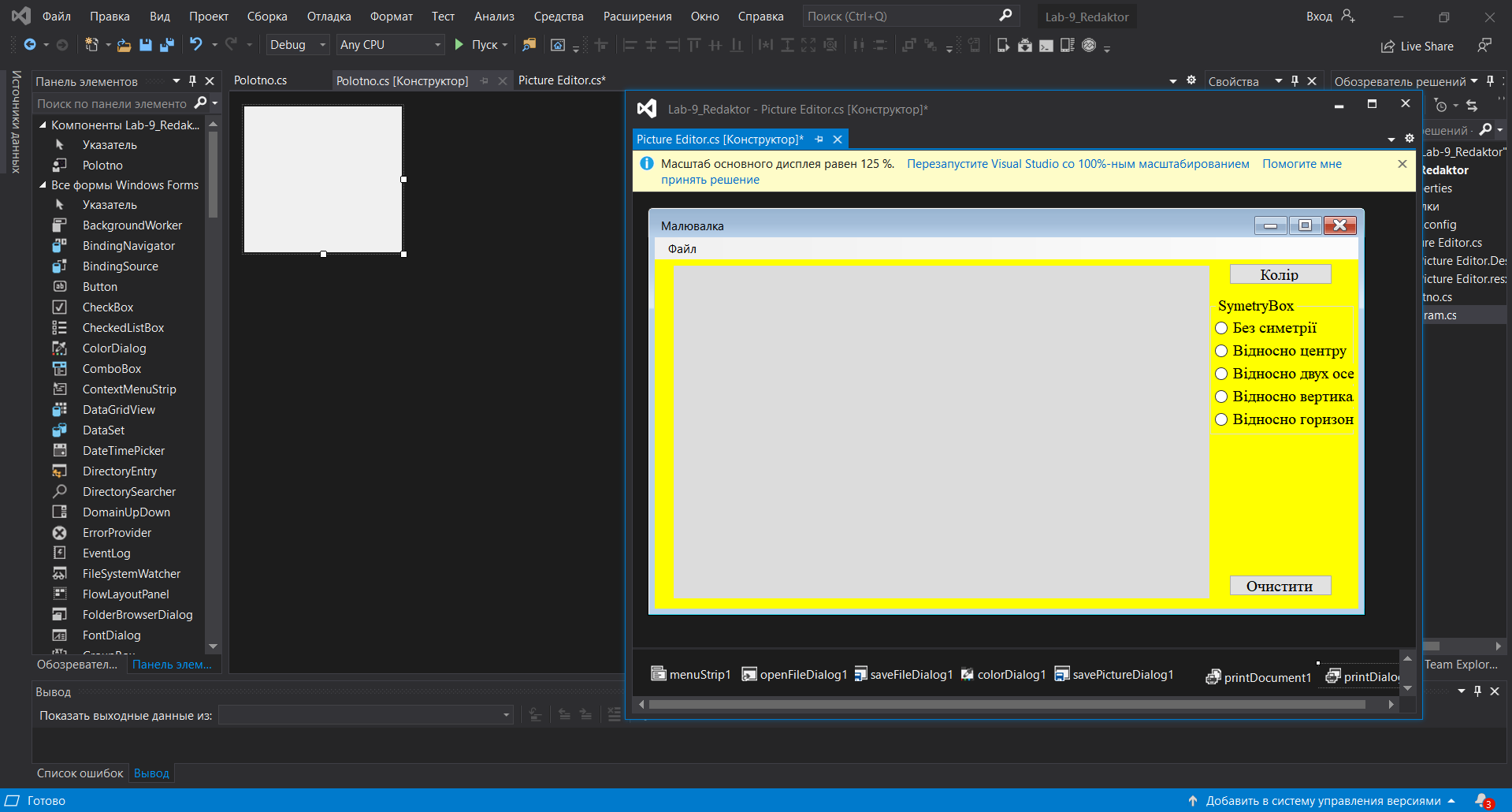
{

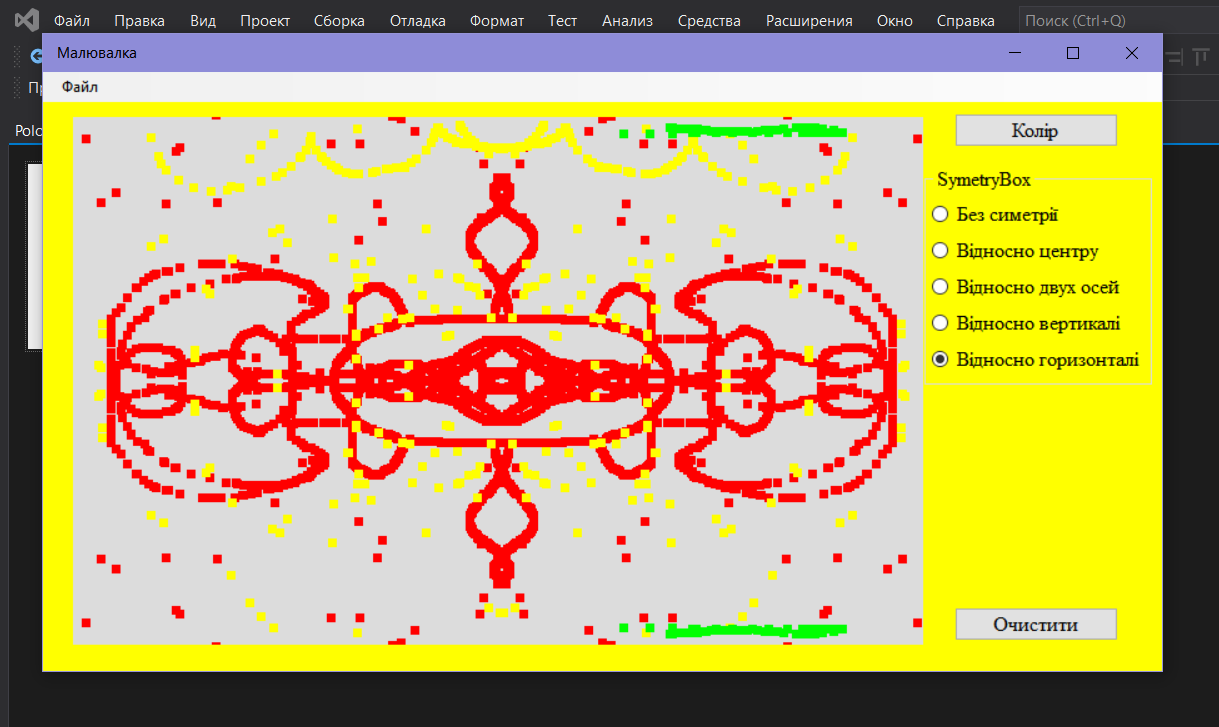
e.Graphics.DrawImage(SavePicture(), 0, 0);

}

}

}

****

****

## Проект «Хрестики-нулики» (xo.exe)

**Постановка задачі:** Розробити програму для гри з комп’ютером у хрестики-нулики.

### Детальний опис функціонування проекту

1. Зовнішній вигляд вікна програми показаний на малюнках:



На початку гри



using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

using System.Collections.Generic;

using System.IO;

namespace TicTacToeForm

{

public partial class TicTacToeForm : Form

{

public TicTacToeForm()

{

InitializeComponent();

MessageBox.Show("Для початку гри зареєструйтеся");

GenerateButtons();

onList();

}

decimal n;

Button[,] buttons = new Button[3, 3];

private string textFromFile1;

private string textFromFile2;

public void onList()

{

string pathToFile = "Winer.txt";

// Считываем строки в массив

string[] allLines = File.ReadAllLines(pathToFile, Encoding.GetEncoding(1251));

// Добавляем каждую строку

foreach (string line in allLines)

listBox1.Items.Add(line);

}

public void Decode()

{

using (FileStream fstream = File.OpenRead($"note.txt"))

{

byte[] array1 = new byte[fstream.Length];

fstream.Read(array1, 0, array1.Length);

textBox1.Text = "Гравець з ніком ";

textBox1.Text+= System.Text.Encoding.Default.GetString(array1);

}

}

public void Winer\_Write()

{

//using (FileStream fstream = new FileStream($"Winer.txt", FileMode.Create))

//{

string writePath = "Winer.txt";

using (StreamWriter sw = new StreamWriter(writePath, true, System.Text.Encoding.Default))

{

sw.WriteLine(textBox1.Text + "Переміг"+"\r\n");

}

//byte[] array1 = System.Text.Encoding.Default.GetBytes(textBox1.Text+" Переміг");

//fstream.Write(array1, 0, array1.Length);

//fstream.Close();

//}

}

public void Decode1()

{

using (FileStream fstream = File.OpenRead($"note1.txt"))

{

byte[] array2 = new byte[fstream.Length];

fstream.Read(array2, 0, array2.Length);

textBox1.Text = "Гравець з ніком ";

textBox1.Text += System.Text.Encoding.Default.GetString(array2);

}

}

private void GenerateButtons()

{

for (int i = 0; i < 3; i++)

{

for (int j = 0; j < 3; j++)

{

buttons[i,j] = new Button();

buttons[i, j].Size = new Size(150, 150);

buttons[i, j].Location = new Point(i \* 150, j \* 150);

buttons[i, j].FlatStyle = FlatStyle.Flat;

buttons[i, j].Font = new System.Drawing.Font(DefaultFont.FontFamily, 80, FontStyle.Bold);

// Define button click event

buttons[i, j].Click += new EventHandler(button\_Click);

// Add button in to the panel

panel1.Controls.Add(buttons[i, j]);

}

}

}

void button\_Click(object sender, EventArgs e)

{

if (i == 1)

{

i = 0;

}

if (i == 2)

{

PlayerButton.Text = "O";

i = 0;

}

button3.Enabled = false;

button3.Visible = false;

Decode();

// Load the clicked button into a local variable

Button button = sender as Button;

// Don't do anything if the block is already marked

if (button.Text != "")

{

return;

}

// Mark the block with current players icon

button.Text = PlayerButton.Text;

if (PlayerButton.Text == "X")

{

button.ForeColor = Color.Green;

}

else

button.ForeColor = Color.Coral;

TogglePlayer();

}

private void TogglePlayer()

{

CheckIfGameEnds();

if (PlayerButton.Text == "X")

{

PlayerButton.Text = "O";

Decode1();

}

else

{

PlayerButton.Text = "X";

Decode();

}

}

private void CheckIfGameEnds()

{

List<Button> winnerButtons = new List<Button>();

#region // vertically

for (int i = 0; i < 3; i++)

{

winnerButtons = new List<Button>();

for (int j = 0; j < 3; j++)

{

if (buttons[i, j].Text != PlayerButton.Text)

{

break;

}

winnerButtons.Add(buttons[i, j]);

if (j == 2)

{

ShowWinner(winnerButtons);

return;

}

}

}

#endregion

#region // horizontally

for (int i = 0; i < 3; i++)

{

winnerButtons = new List<Button>();

for (int j = 0; j < 3; j++)

{

if (buttons[j, i].Text != PlayerButton.Text)

{

break;

}

winnerButtons.Add(buttons[j, i]);

if (j == 2)

{

ShowWinner(winnerButtons);

return;

}

}

}

#endregion

#region// diagonally 1 (top-left to bottom-right)

winnerButtons = new List<Button>();

for (int i = 0, j = 0; i < 3; i++, j++)

{

if (buttons[i, j].Text != PlayerButton.Text)

{

break;

}

winnerButtons.Add(buttons[i, j]);

if (j == 2)

{

ShowWinner(winnerButtons);

return;

}

}

#endregion

#region// diagonally 2 (bottom-left to top-right)

winnerButtons = new List<Button>();

for (int i = 2, j = 0; j < 3; i--, j++)

{

if (buttons[i, j].Text != PlayerButton.Text)

{

break;

}

winnerButtons.Add(buttons[i, j]);

if (j == 2)

{

ShowWinner(winnerButtons);

return;

}

}

#endregion

// check if all the blocks are marked

foreach (var button in buttons)

{

if (button.Text == "")

return;

}

MessageBox.Show("Гра закінчилася нічиєю");

Application.Restart();

}

private void ShowWinner(List<Button> winnerButtons)

{

// color all the winner blocks

foreach (var button in winnerButtons)

{

button.BackColor = Color.Red;

}

MessageBox.Show("Гравець " + PlayerButton.Text + " ПЕРЕМІГ");

Winer\_Write();

Application.Restart();

}

private void button1\_Click(object sender, EventArgs e)

{

button1.Enabled = false;

button1.Visible = false;

button2.Enabled = false;

button2.Visible = false;

Reg reg = new Reg();

reg.Show();

}

private void button2\_Click(object sender, EventArgs e)

{

button1.Enabled = false;

button1.Visible = false;

button2.Enabled = false;

button2.Visible = false;

}

int i = 0;

private void button3\_Click(object sender, EventArgs e)

{

Random random = new Random();

int value = random.Next(1, 3);

if (value == 1)

{

MessageBox.Show("Перший ходить гравець Х");

i = 1;

}

else

{

MessageBox.Show("Перший ходить гравець O");

i = 2;

}

button3.Enabled = false;

button3.Visible = false;

if (i == 2)

{

PlayerButton.Text = "O";

i = 0;

}

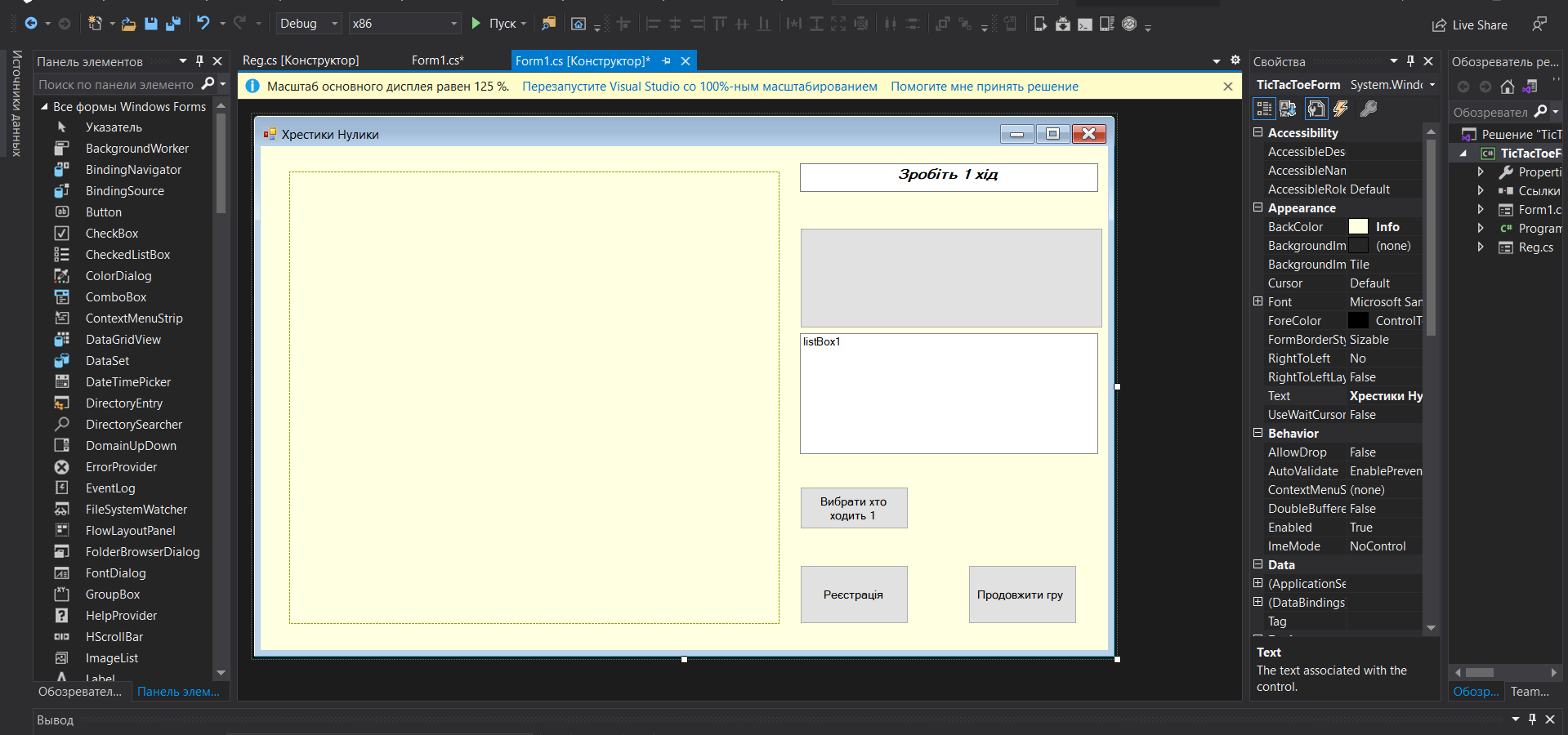
else PlayerButton.Text = "X";

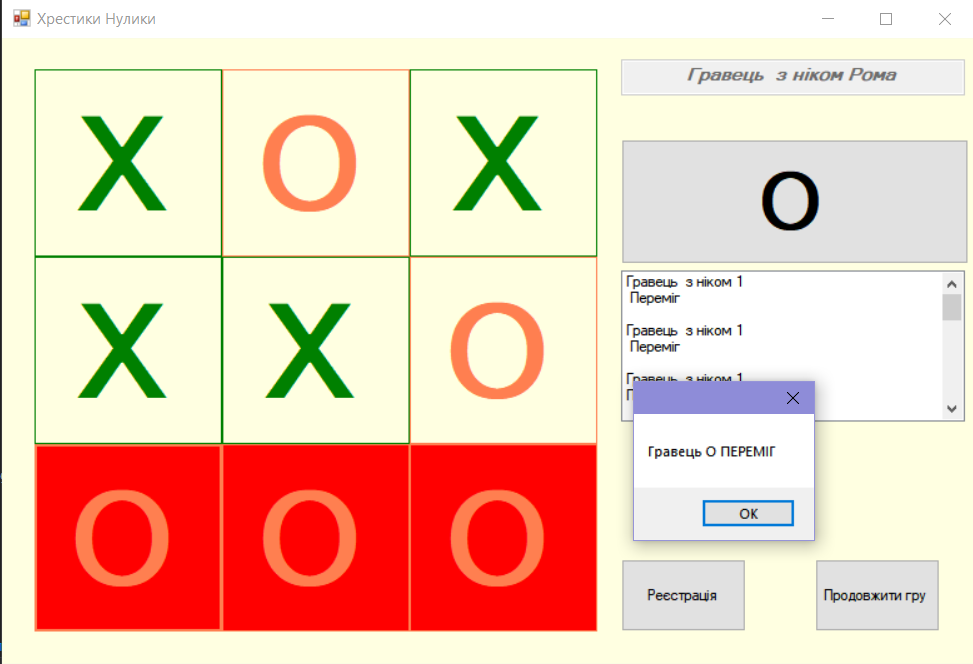
panel1.Enabled = true;

}

}

}





Висновок: На даній лабораторній роботі я одержав практичні по створенню нащадків класів та компонентів в C#.