Министерство образования Республики Беларусь

Оршанский колледж ВГУ имени П.М.Машерова

**Отчет**

по учебной практике по программированию

«Модуль 5. Графический интерфейс»

по предмету «Конструирование программ и языки программирования»

Выполнил учащийся Казаченко Н.С.

группы 3ПОИС23 02.10.2025 г.

Проверил Алейников М.А.

02.10.2025 г.

Орша, 2025

**Цель:** закрепить знания по графическим интерфейсам на ЯП С#.

**Модуль 5 Графический интерфейс:**

1. Создайте графическое приложение для рисования на холсте с использованием Windows Forms. Реализуйте функции рисования линий, кругов и квадратов.
2. Разработайте текстовый редактор с возможностью открытия и сохранения текстовых файлов.
3. Создайте приложение для учета задач с использованием Windows Forms. Пользователь должен иметь возможность добавлять, удалять и отмечать задачи как выполненные.
4. Реализуйте приложение для просмотра изображений. Пользователь должен иметь возможность выбирать изображение для просмотра и масштабировать его.
5. Создайте калькулятор с графическим интерфейсом. Пользователь должен иметь возможность выполнять арифметические операции.

**Ход работы**

**Выполнение задания 1.**

**Листинг 1.** MainWindow.xaml

<Window x:Class="Задание\_1.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:Задание\_1"

mc:Ignorable="d"

Title="Рисовалка" Height="600" Width="800" Background="LightGray">

<Grid>

<!-- Панель инструментов -->

<StackPanel Orientation="Horizontal" VerticalAlignment="Top" Background="DarkSlateGray" Height="50">

<Button Name="LineButton" Content="Линия" Width="100" Margin="5" Background="LightSteelBlue" Foreground="White"/>

<Button Name="CircleButton" Content="Круг" Width="100" Margin="5" Background="LightSteelBlue" Foreground="White"/>

<Button Name="SquareButton" Content="Квадрат" Width="100" Margin="5" Background="LightSteelBlue" Foreground="White"/>

<TextBlock Text="Выберите форму и нарисуйте на холсте" VerticalAlignment="Center" Margin="10,0,0,0" Foreground="White"/>

</StackPanel>

<!-- Холст для рисования -->

<Canvas Name="DrawingCanvas" Background="WhiteSmoke" Margin="10,60,10,10" />

</Grid>

</Window>

**Листинг 2.** MainWindow.xaml.cs

using System;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Shapes;

namespace Задание\_1

{

public partial class MainWindow : Window

{

private enum ShapeType { None, Line, Circle, Square }

private ShapeType currentShape = ShapeType.None;

private Point startPoint;

public MainWindow()

{

InitializeComponent();

LineButton.Click += LineButton\_Click;

CircleButton.Click += CircleButton\_Click;

SquareButton.Click += SquareButton\_Click;

DrawingCanvas.MouseDown += DrawingCanvas\_MouseDown;

DrawingCanvas.MouseUp += DrawingCanvas\_MouseUp;

}

private void LineButton\_Click(object sender, RoutedEventArgs e)

{

currentShape = ShapeType.Line;

}

private void CircleButton\_Click(object sender, RoutedEventArgs e)

{

currentShape = ShapeType.Circle;

}

private void SquareButton\_Click(object sender, RoutedEventArgs e)

{

currentShape = ShapeType.Square;

}

private void DrawingCanvas\_MouseDown(object sender, MouseButtonEventArgs e)

{

startPoint = e.GetPosition(DrawingCanvas);

}

private void DrawingCanvas\_MouseUp(object sender, MouseButtonEventArgs e)

{

Point endPoint = e.GetPosition(DrawingCanvas);

switch (currentShape)

{

case ShapeType.Line:

DrawLine(startPoint, endPoint);

break;

case ShapeType.Circle:

DrawCircle(startPoint, endPoint);

break;

case ShapeType.Square:

DrawSquare(startPoint, endPoint);

break;

}

}

private void DrawLine(Point start, Point end)

{

Line line = new Line

{

X1 = start.X,

Y1 = start.Y,

X2 = end.X,

Y2 = end.Y,

Stroke = Brushes.Black,

StrokeThickness = 2

};

DrawingCanvas.Children.Add(line);

}

private void DrawCircle(Point start, Point end)

{

double radius = Math.Abs(end.X - start.X);

Ellipse circle = new Ellipse

{

Width = radius \* 2,

Height = radius \* 2,

Stroke = Brushes.Blue,

StrokeThickness = 2

};

Canvas.SetLeft(circle, start.X - radius);

Canvas.SetTop(circle, start.Y - radius);

DrawingCanvas.Children.Add(circle);

}

private void DrawSquare(Point start, Point end)

{

double side = Math.Abs(end.X - start.X);

Rectangle square = new Rectangle

{

Width = side,

Height = side,

Stroke = Brushes.Green,

StrokeThickness = 2

};

Canvas.SetLeft(square, start.X);

Canvas.SetTop(square, start.Y);

DrawingCanvas.Children.Add(square);

}

}

}

На рисунке 1 представлена диаграмма классов программы.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 1 – Диаграмма классов

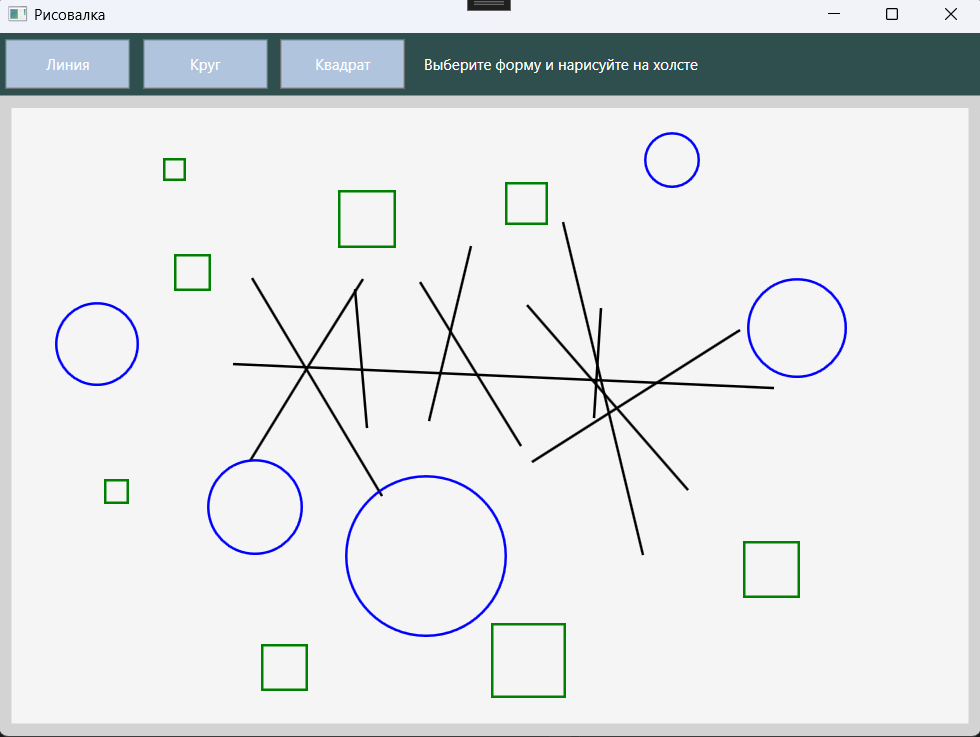
На рисунке 2 представлено выполнение задания создания графического приложения для рисования на холсте с использованием Windows Forms. 

Рисунок 2 – Работа программы

**Выполнение задания 2.**

**Листинг 3.** MainWindow.xaml

using System;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Shapes;

namespace Задание\_1

{

public partial class MainWindow : Window

{

private enum ShapeType { None, Line, Circle, Square }

private ShapeType currentShape = ShapeType.None;

private Point startPoint;

public MainWindow()

{

InitializeComponent();

LineButton.Click += LineButton\_Click;

CircleButton.Click += CircleButton\_Click;

SquareButton.Click += SquareButton\_Click;

DrawingCanvas.MouseDown += DrawingCanvas\_MouseDown;

DrawingCanvas.MouseUp += DrawingCanvas\_MouseUp;

}

private void LineButton\_Click(object sender, RoutedEventArgs e)

{

currentShape = ShapeType.Line;

}

private void CircleButton\_Click(object sender, RoutedEventArgs e)

{

currentShape = ShapeType.Circle;

}

private void SquareButton\_Click(object sender, RoutedEventArgs e)

{

currentShape = ShapeType.Square;

}

private void DrawingCanvas\_MouseDown(object sender, MouseButtonEventArgs e)

{

startPoint = e.GetPosition(DrawingCanvas);

}

private void DrawingCanvas\_MouseUp(object sender, MouseButtonEventArgs e)

{

Point endPoint = e.GetPosition(DrawingCanvas);

switch (currentShape)

{

case ShapeType.Line:

DrawLine(startPoint, endPoint);

break;

case ShapeType.Circle:

DrawCircle(startPoint, endPoint);

break;

case ShapeType.Square:

DrawSquare(startPoint, endPoint);

break;

}

}

private void DrawLine(Point start, Point end)

{

Line line = new Line

{

X1 = start.X,

Y1 = start.Y,

X2 = end.X,

Y2 = end.Y,

Stroke = Brushes.Black,

StrokeThickness = 2

};

DrawingCanvas.Children.Add(line);

}

private void DrawCircle(Point start, Point end)

{

double radius = Math.Abs(end.X - start.X);

Ellipse circle = new Ellipse

{

Width = radius \* 2,

Height = radius \* 2,

Stroke = Brushes.Blue,

StrokeThickness = 2

};

Canvas.SetLeft(circle, start.X - radius);

Canvas.SetTop(circle, start.Y - radius);

DrawingCanvas.Children.Add(circle);

}

private void DrawSquare(Point start, Point end)

{

double side = Math.Abs(end.X - start.X);

Rectangle square = new Rectangle

{

Width = side,

Height = side,

Stroke = Brushes.Green,

StrokeThickness = 2

};

Canvas.SetLeft(square, start.X);

Canvas.SetTop(square, start.Y);

DrawingCanvas.Children.Add(square);

}

}

}

**Листинг 4.** MainWindow.xaml.cs

using Microsoft.Win32;

using System.Windows;

using TextEditorApp;

namespace Задание\_2

{

public partial class MainWindow : Window

{

private readonly FileManager fileManager = new FileManager();

public MainWindow()

{

InitializeComponent();

}

public FileManager FileManager

{

get => default;

set

{

}

}

private void OpenButton\_Click(object sender, RoutedEventArgs e)

{

OpenFileDialog openFileDialog = new OpenFileDialog

{

Filter = "Text Files (\*.txt)|\*.txt|All Files (\*.\*)|\*.\*"

};

if (openFileDialog.ShowDialog() == true)

{

string fileContent = fileManager.OpenFile(openFileDialog.FileName);

EditorTextBox.Text = fileContent;

}

}

private void SaveButton\_Click(object sender, RoutedEventArgs e)

{

SaveFileDialog saveFileDialog = new SaveFileDialog

{

Filter = "Text Files (\*.txt)|\*.txt|All Files (\*.\*)|\*.\*"

};

if (saveFileDialog.ShowDialog() == true)

{

fileManager.SaveFile(saveFileDialog.FileName, EditorTextBox.Text);

}

}

}

}

**Листинг 5.** FileManager.cs

using System.IO;

namespace TextEditorApp

{

public class FileManager

{

// Метод для открытия файла и чтения его содержимого

public string OpenFile(string filePath)

{

if (File.Exists(filePath))

{

return File.ReadAllText(filePath);

}

return string.Empty;

}

// Метод для сохранения данных в файл

public void SaveFile(string filePath, string content)

{

File.WriteAllText(filePath, content);

}

}

}

На рисунке 3 представлена диаграмма классов программы.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 3 – Диаграмма классов

На рисунке 4 представлено выполнение задания создания приложения текстового редактора с графическим интерфейсом.

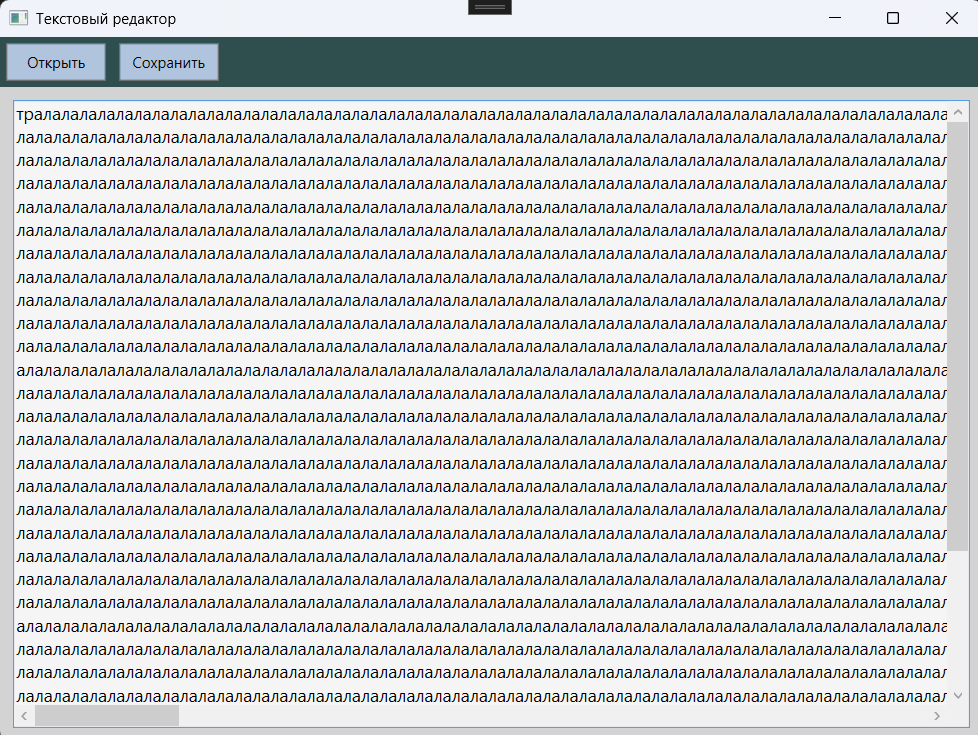


Рисунок 4 – Работа программы

**Выполнение задания 3.**

**Листинг 6.** MainWindow.xaml

<Window x:Class="Задание\_3.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:Задание\_3"

mc:Ignorable="d"

Title="Задачи" Height="400" Width="400" Background="LightGray">

<DockPanel>

<!-- Панель для добавления новой задачи -->

<StackPanel DockPanel.Dock="Top" Orientation="Horizontal" Margin="10">

<TextBox Name="TaskInputTextBox" Width="250" Margin="5" />

<Button Name="AddTaskButton" Content="Добавить задачу" Width="100" Margin="5"

Click="AddTaskButton\_Click" Background="LightSteelBlue" Foreground="Black"/>

</StackPanel>

<!-- Список задач -->

<ListBox Name="TasksListBox" Margin="10" SelectionChanged="TasksListBox\_SelectionChanged">

<ListBox.ItemTemplate>

<DataTemplate>

<StackPanel Orientation="Horizontal">

<CheckBox IsChecked="{Binding IsCompleted}" Content="{Binding Description}" Margin="5"/>

<Button Content="Удалить" Width="75" Margin="5" Click="DeleteButton\_Click"/>

</StackPanel>

</DataTemplate>

</ListBox.ItemTemplate>

</ListBox>

</DockPanel>

</Window>

**Листинг 7.** MainWindow.xaml.cs

using System.Windows;

using System.Windows.Controls;

using TaskManagerApp;

namespace Задание\_3

{

public partial class MainWindow : Window

{

private readonly TaskManager \_taskManager = new TaskManager();

public MainWindow()

{

InitializeComponent();

TasksListBox.ItemsSource = \_taskManager.Tasks; // Устанавливаем источник данных для списка задач

}

private void AddTaskButton\_Click(object sender, RoutedEventArgs e)

{

string taskDescription = TaskInputTextBox.Text.Trim();

if (!string.IsNullOrEmpty(taskDescription))

{

\_taskManager.AddTask(taskDescription);

TaskInputTextBox.Clear(); // Очищаем текстовое поле после добавления задачи

}

}

private void DeleteButton\_Click(object sender, RoutedEventArgs e)

{

if (sender is Button button)

{

TaskItem task = button.DataContext as TaskItem;

if (task != null)

{

\_taskManager.RemoveTask(task);

}

}

}

private void TasksListBox\_SelectionChanged(object sender, System.Windows.Controls.SelectionChangedEventArgs e)

{

}

}

}

**Листинг 8.** TaskItem.cs

using System.ComponentModel;

namespace TaskManagerApp

{

public class TaskItem : INotifyPropertyChanged

{

private bool \_isCompleted;

public string Description { get; set; }

public bool IsCompleted

{

get => \_isCompleted;

set

{

\_isCompleted = value;

OnPropertyChanged(nameof(IsCompleted));

}

}

public event PropertyChangedEventHandler PropertyChanged;

protected void OnPropertyChanged(string propertyName)

{

PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propertyName));

}

}

}

**Листинг 9.** TaskManager.cs

using System.Collections.ObjectModel;

namespace TaskManagerApp

{

public class TaskManager

{

public ObservableCollection<TaskItem> Tasks { get; set; }

public TaskManager()

{

Tasks = new ObservableCollection<TaskItem>();

}

public void AddTask(string description)

{

Tasks.Add(new TaskItem { Description = description, IsCompleted = false });

}

public void RemoveTask(TaskItem task)

{

Tasks.Remove(task);

}

}

}

На рисунке 5 представлена диаграмма классов программы.

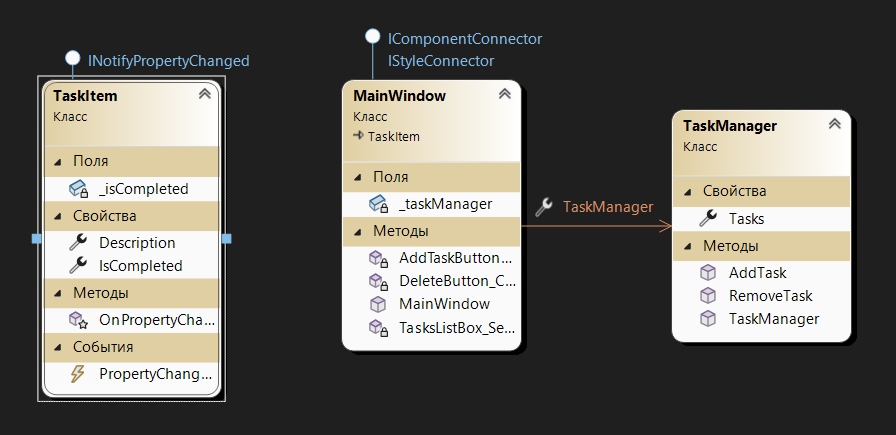


Рисунок 5 – Диаграмма классов

На рисунке 6 представлено выполнение задания создания приложения для учёта задач с использованием Windows Forms.

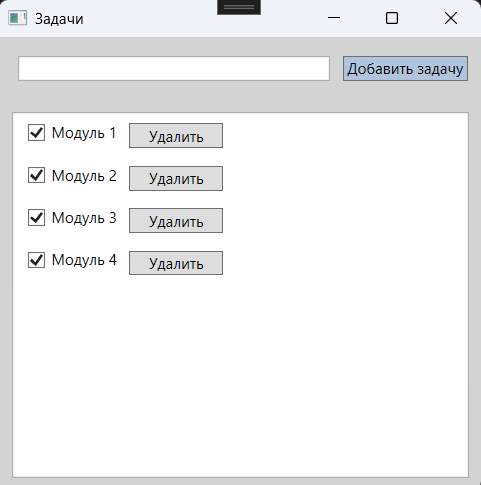


Рисунок 6 – Работа программы

**Выполнение задания 4.**

**Листинг 10.** MainWindow.xaml

<Window x:Class="Задание\_4.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:Задание\_4"

mc:Ignorable="d"

Title="Средство просмотра изображений" Height="600" Width="800">

<Grid>

<Menu VerticalAlignment="Top">

<MenuItem Header="Файл">

<MenuItem Header="Открыть" Click="OpenMenuItem\_Click" />

</MenuItem>

</Menu>

<Image Name="ImageView" Stretch="Uniform" HorizontalAlignment="Center" VerticalAlignment="Center"/>

<Slider Name="ZoomSlider" Minimum="0.1" Maximum="5.0" Value="1.0"

Width="200" Margin="10" VerticalAlignment="Bottom"

ValueChanged="ZoomSlider\_ValueChanged"/>

<TextBlock Text="Увеличение" Margin="220,0,0,0"

VerticalAlignment="Bottom" HorizontalAlignment="Left"

FontWeight="Bold" FontSize="14"/>

</Grid>

</Window>

**Листинг 11.** MainWindow.xaml.cs

using Microsoft.Win32;

using System;

using System.Windows;

using System.Windows.Media.Imaging;

namespace Задание\_4

{

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

}

private void OpenMenuItem\_Click(object sender, RoutedEventArgs e)

{

OpenFileDialog openFileDialog = new OpenFileDialog

{

Filter = "Images (\*.jpg;\*.jpeg;\*.png;\*.bmp)|\*.jpg;\*.jpeg;\*.png;\*.bmp"

};

if (openFileDialog.ShowDialog() == true)

{

BitmapImage bitmap = new BitmapImage(new Uri(openFileDialog.FileName));

ImageView.Source = bitmap; // Установка изображения в элемент Image

ZoomSlider.Value = 1; // Сброс значения слайдера при открытии нового изображения

}

}

private void ZoomSlider\_ValueChanged(object sender, RoutedPropertyChangedEventArgs<double> e)

{

if (ImageView.Source != null)

{

ImageView.Width = ImageView.Source.Width \* ZoomSlider.Value; // Изменение ширины

ImageView.Height = ImageView.Source.Height \* ZoomSlider.Value; // Изменение высоты

}

}

}

}

На рисунке 7 представлена диаграмма классов программы.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 7 – Диаграмма классов

На рисунке 8 представлено выполнение задания создания приложения для просмотра изображений с использованием Windows Forms.

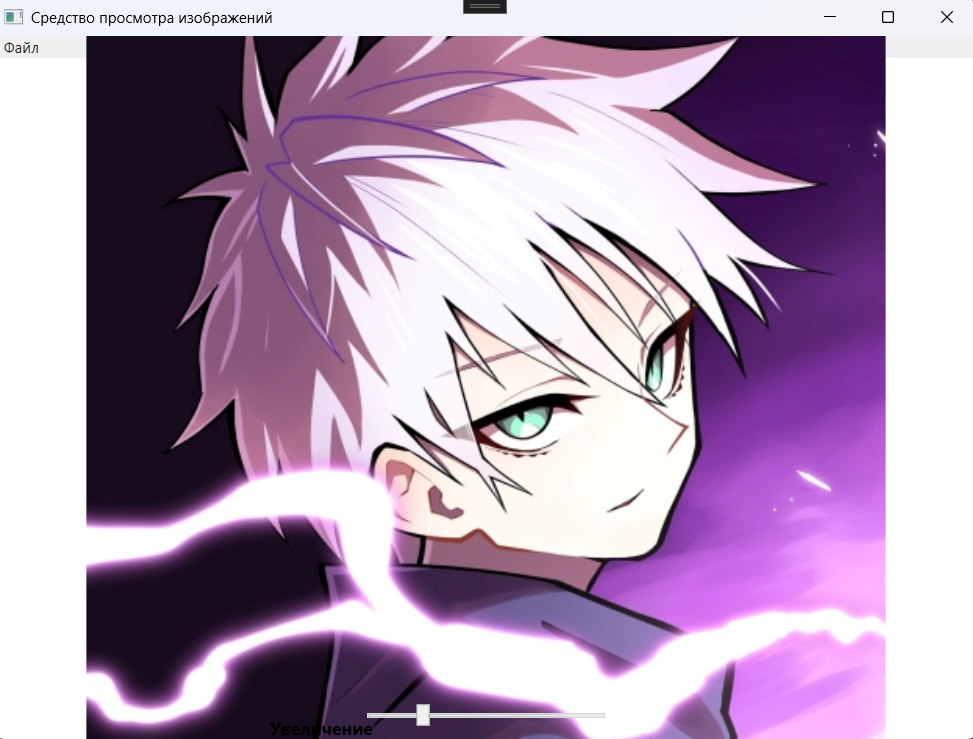


Рисунок 8 – Работа программы

**Выполнение задания 5.**

**Листинг 12.** MainWindow.xaml

<Window x:Class="Задание\_5.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:Задание\_5"

mc:Ignorable="d"

Title="Калькулятор" Height="350" Width="300">

<Grid>

<TextBox Name="ResultTextBox" Height="50" FontSize="24" Margin="10" VerticalAlignment="Top" IsReadOnly="True" HorizontalAlignment="Stretch"/>

<UniformGrid Rows="4" Columns="4" Margin="10,70,10,10">

<Button Content="7" Click="Button\_Click"/>

<Button Content="8" Click="Button\_Click"/>

<Button Content="9" Click="Button\_Click"/>

<Button Content="/" Click="Operator\_Click"/>

<Button Content="4" Click="Button\_Click"/>

<Button Content="5" Click="Button\_Click"/>

<Button Content="6" Click="Button\_Click"/>

<Button Content="\*" Click="Operator\_Click"/>

<Button Content="1" Click="Button\_Click"/>

<Button Content="2" Click="Button\_Click"/>

<Button Content="3" Click="Button\_Click"/>

<Button Content="-" Click="Operator\_Click"/>

<Button Content="0" Click="Button\_Click"/>

<Button Content="C" Click="Clear\_Click"/>

<Button Content="=" Click="Equals\_Click"/>

<Button Content="+" Click="Operator\_Click"/>

</UniformGrid>

</Grid>

</Window>

**Листинг 13.** MainWindow.xaml.cs

using System.Windows;

using System.Windows.Controls;

namespace Задание\_5

{

public partial class MainWindow : Window

{

private double \_result;

private string \_operation;

private bool \_isOperationPerformed;

public MainWindow()

{

InitializeComponent();

}

private void Button\_Click(object sender, RoutedEventArgs e)

{

Button button = (Button)sender;

if (\_isOperationPerformed)

{

ResultTextBox.Text = button.Content.ToString();

\_isOperationPerformed = false;

}

else

{

ResultTextBox.Text += button.Content.ToString();

}

}

private void Operator\_Click(object sender, RoutedEventArgs e)

{

Button button = (Button)sender;

if (double.TryParse(ResultTextBox.Text, out \_result))

{

\_operation = button.Content.ToString();

\_isOperationPerformed = true;

}

}

private void Equals\_Click(object sender, RoutedEventArgs e)

{

double secondOperand;

if (double.TryParse(ResultTextBox.Text, out secondOperand))

{

switch (\_operation)

{

case "+":

\_result += secondOperand;

break;

case "-":

\_result -= secondOperand;

break;

case "\*":

\_result \*= secondOperand;

break;

case "/":

if (secondOperand != 0)

\_result /= secondOperand;

else

ResultTextBox.Text = "Error";

break;

}

ResultTextBox.Text = \_result.ToString();

\_operation = string.Empty;

}

}

private void Clear\_Click(object sender, RoutedEventArgs e)

{

ResultTextBox.Text = string.Empty;

\_result = 0;

\_operation = string.Empty;

}

}

}

На рисунке 9 представлена диаграмма классов программы.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 9 – Диаграмма классов

На рисунке 10 представлено выполнение задания создания приложения калькулятора с графическим интерфейсом.

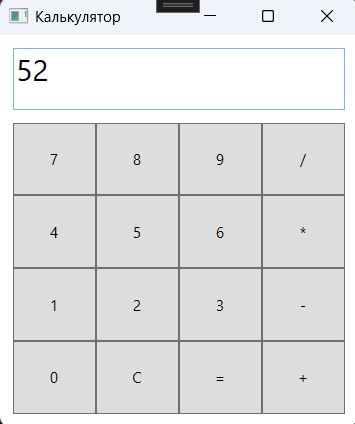


Рисунок 10 – Работа программы

**Ссылка на репозиторий:**

[**https://github.com/Seyko00/PraktikaPoProgrammirovaniu/tree/master/%D0%9C%D0%BE%D0%B4%D1%83%D0%BB%D1%8C%201.1**](https://github.com/Seyko00/PraktikaPoProgrammirovaniu/tree/master/%D0%9C%D0%BE%D0%B4%D1%83%D0%BB%D1%8C%201.1)

**Вывод:** в ходе выполнения практического задания модуля 5 были закреплены знания по графическим интерфейсам на языке программирования С#.