ГУАП

ФАКУЛЬТЕТ СРЕДНЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| преподаватель |  |  |  | И.А. Юрьева |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |
| ОТЧЕТЫ О ЛАБОРАТОРНЫХ РАБОТАХ | | | | |
|  | | | | |
| по дисциплине: МДК 01.01 | | | | |
|  | | | | |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. № | С021 |  |  |  | Гамуйло С.С. |
|  |  |  | подпись, дата |  | инициалы, фамилия |

СОДЕРЖАНИЕ

[Лабораторная работа №1 3](#_Toc117936163)

[Лабораторная работа №2 10](#_Toc117936164)

[Лабораторная работа №3 15](#_Toc117936165)

# Лабораторная работа №1

**Тема:** Создание приложения «Микропроводник».

**Цель работы**: получение практических навыков при работе с пространством имен **System.IO.**

**Задание 1.** Разработать приложение «Микропроводник», примерный вид которого представлен на Рисунке 1.

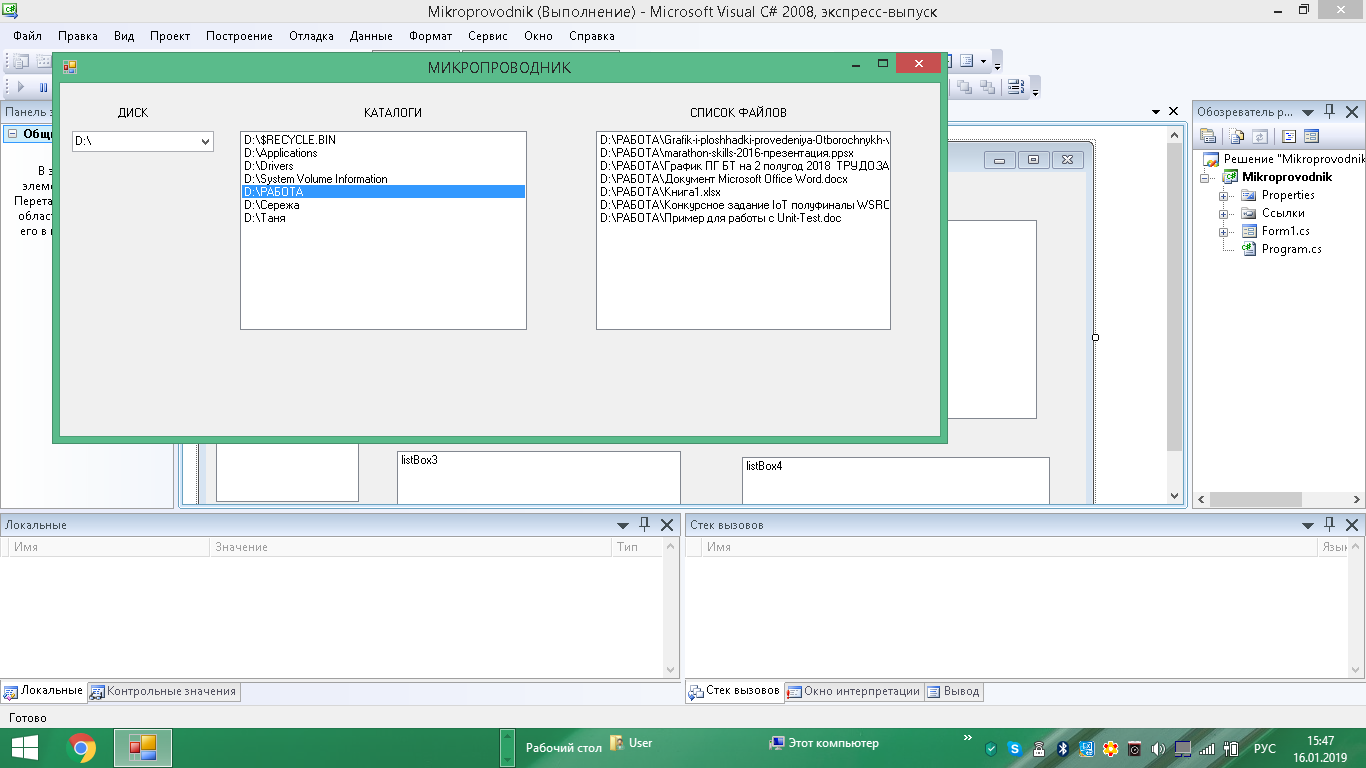


Рисунок 1 – Приложение Микропроводник

На форме список всех дисков загружается в компонент comboBox1. Список всех каталогов для данного диска загружается в listBox1. Список файлов, находящихся в выбранном каталоге, отображается listBox2.

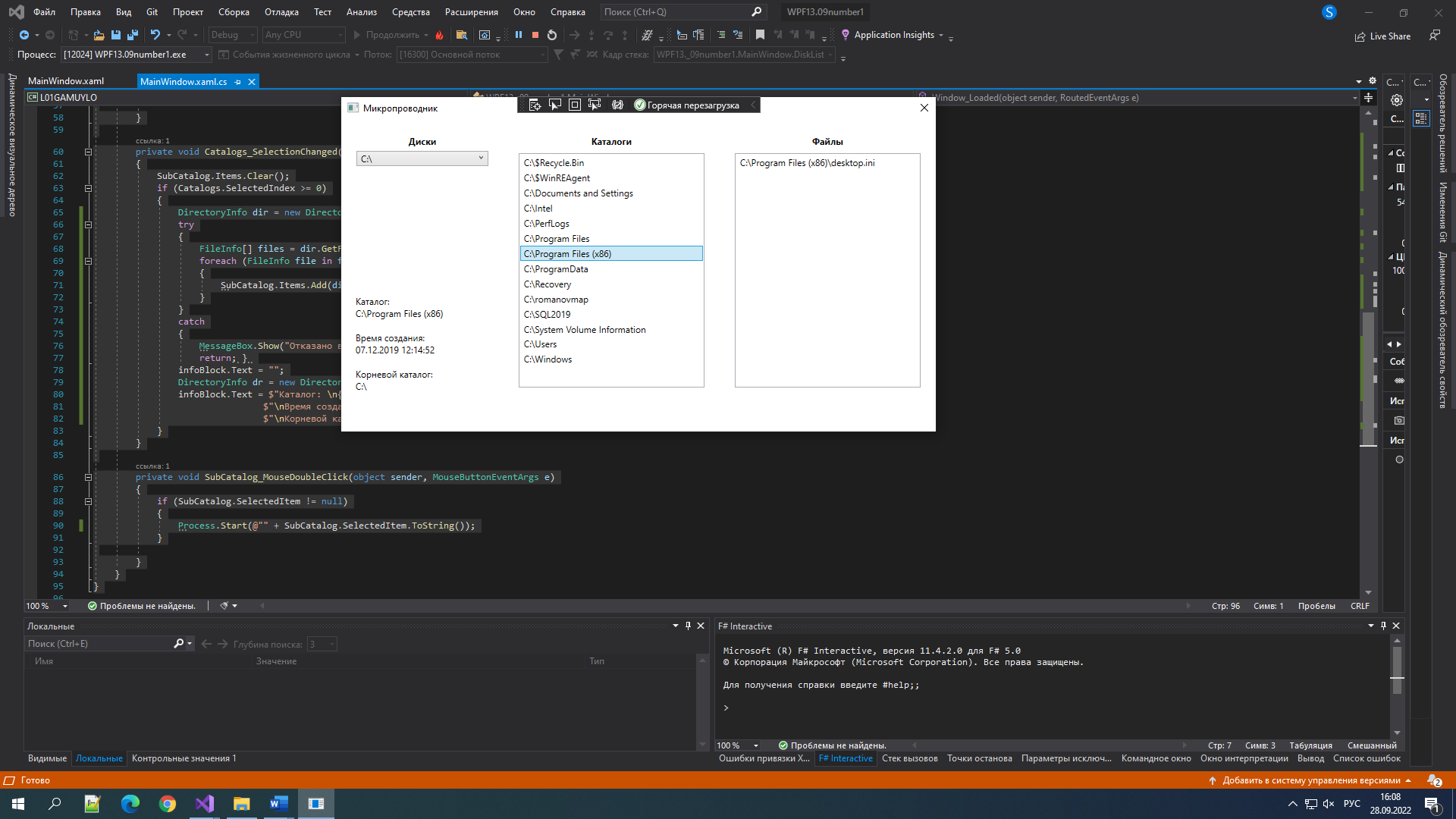
**Задание 2.** Используя дополнительные компоненты,

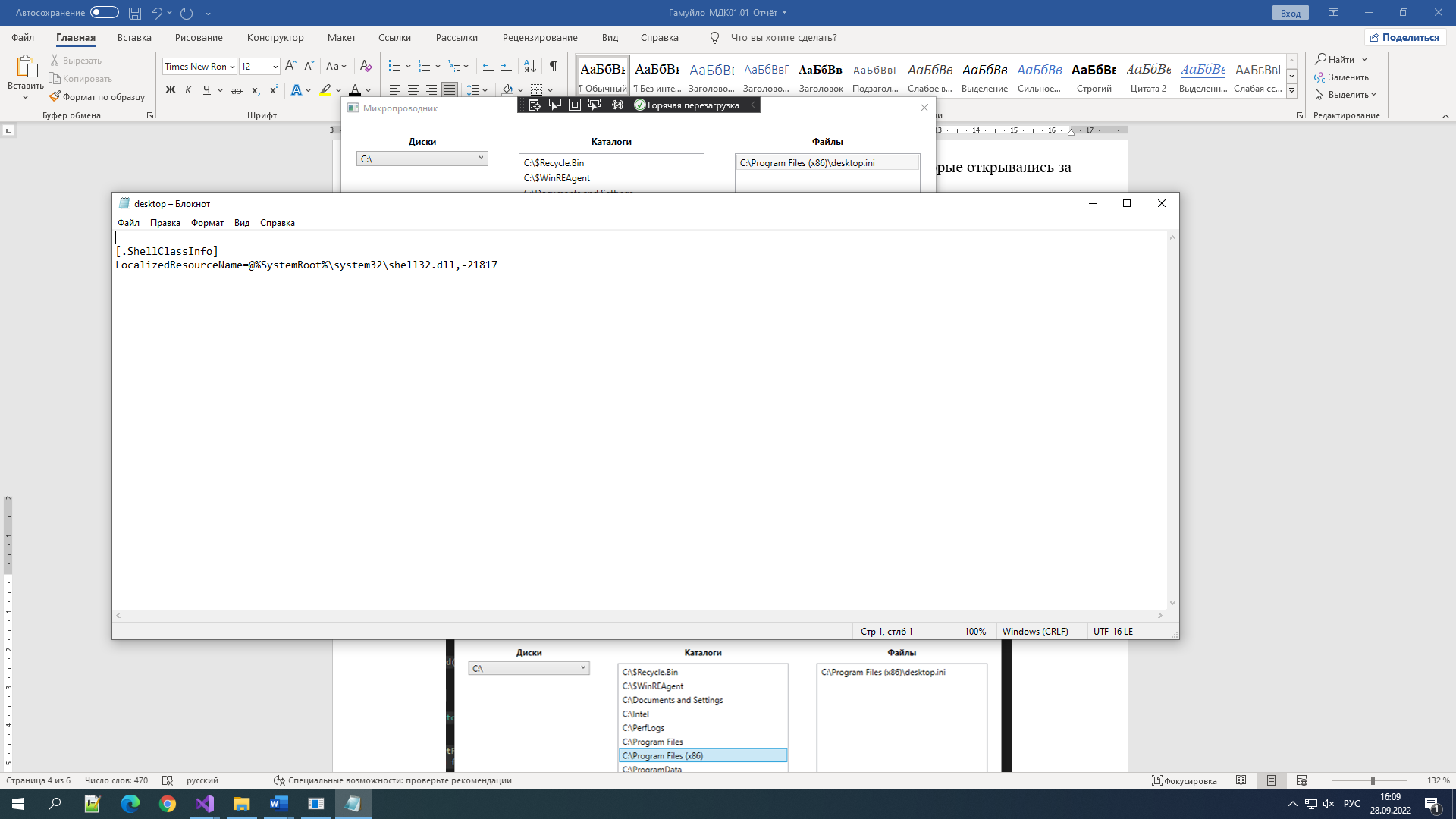
* для выделенного диска необходимо выводить сведения: объем диска, свободное пространство;
* для выделенного каталога: полное название каталога, время создания каталога, корневой каталог.

**Задание 3.** При выделении файла в списке должно запускаться соответствующее приложение.

**Задание 4.** Сохранить в отдельный текстовый файл имена файлов, которые открывались за последние 10 секунд работы приложения.

Скриншоты результатов работы программы





Листинги

XAML-разметка:

<Window x:Class="WPF13.\_09number1.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:WPF13.\_09number1"

mc:Ignorable="d"

Loaded="Window\_Loaded"

ResizeMode="NoResize"

Title="Микропроводник" Height="450" Width="800">

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="3\*"/>

<ColumnDefinition Width="4\*"/>

<ColumnDefinition Width="4\*"/>

</Grid.ColumnDefinitions>

<TextBlock FontWeight="Bold" Grid.Column="1" Text="Каталоги" HorizontalAlignment="Center" Margin="111,20,111,375"></TextBlock>

<TextBlock FontWeight="Bold" Grid.Column="2" Text="Файлы" HorizontalAlignment="Center" Margin="90,20,90,375"></TextBlock>

<TextBlock FontWeight="Bold" Grid.Column="0" Text="Диски" HorizontalAlignment="Center" Margin="89,20,89,375"></TextBlock>

<ComboBox SelectionChanged="DiskList\_Selected" Name="DiskList" Grid.Column="0" Height="20" Margin="20,40,20,350">

</ComboBox>

<ListBox SelectionChanged="Catalogs\_SelectionChanged" Name="Catalogs" Grid.Column="1" Margin="20,44,20,58">

</ListBox>

<ListBox Name="SubCatalog" MouseDoubleClick="SubCatalog\_MouseDoubleClick" Grid.Column="2" Margin="20,44,20,58">

</ListBox>

<TextBlock Padding="20" TextWrapping="Wrap" x:Name="infoBlock" VerticalAlignment="Bottom" HorizontalAlignment="Center" Width="216" Height="200"/>

</Grid>

</Window>

Обработчики событий:

using System;

using System.Collections.Generic;

using System.Collections;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

using System.IO;

using System.Diagnostics;

using System.Timers;

using System.Threading;

using System.Windows.Threading;

namespace WPF13.\_09number1

{

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

}

private void Window\_Loaded(object sender, RoutedEventArgs e)

{

DriveInfo[] drives = DriveInfo.GetDrives();

foreach (DriveInfo drive in drives)

{

DiskList.Items.Add(drive.Name);

}

}

private void DiskList\_Selected(object sender, RoutedEventArgs e)

{

{

infoBlock.Text = "";

SubCatalog.Items.Clear();

Catalogs.Items.Clear();

DriveInfo[] drives = DriveInfo.GetDrives();

DriveInfo disk = new DriveInfo(DiskList.SelectedItem.ToString());

DirectoryInfo directories = new DirectoryInfo(DiskList.SelectedItem.ToString());

foreach (var d in directories.GetDirectories())

{

Catalogs.Items.Add(directories + d.Name);

}

infoBlock.Text = $"Название диска: \n{disk} \n" +

$"\nОбъём диска: \n{Math.Round(((Convert.ToDouble(disk.TotalSize) / 1024) / 1024)/ 1024, 1)} гигабайт \n" +

$"\nОбъем свободного места: \n{Math.Round(((Convert.ToDouble(disk.TotalFreeSpace) / 1024) / 1024)/ 1024, 1)} гигабайт";

}

}

private void Catalogs\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

SubCatalog.Items.Clear();

if (Catalogs.SelectedIndex >= 0)

{

DirectoryInfo dir = new DirectoryInfo(Catalogs.SelectedItem.ToString());

try

{

FileInfo[] files = dir.GetFiles();

foreach (FileInfo file in files)

{

SubCatalog.Items.Add(dir + "\\" + file.Name);

}

}

catch

{

MessageBox.Show("Отказано в доступе");

return; }

infoBlock.Text = "";

DirectoryInfo dr = new DirectoryInfo(Catalogs.SelectedItem.ToString());

infoBlock.Text = $"Каталог: \n{dr.FullName} \n" +

$"\nВремя создания: \n{dr.CreationTime} \n" +

$"\nКорневой каталог: \n{dr.Root}";

}

}

private void SubCatalog\_MouseDoubleClick(object sender, MouseButtonEventArgs e)

{

if (SubCatalog.SelectedItem != null)

{

Process.Start(@"" + SubCatalog.SelectedItem.ToString());

}

}

}

# Лабораторная работа №2

Тема: Дополнительные задания на работу с файловой системой

**Задание 1 (фрагмент модуля задания по компетенции Программные решения для бизнеса(2018 год)).**Вы являетесь разработчиком в команде, которая занимается проектированием и разработкой настольных приложений, взаимодействующих с БД. Технические сложности пока не дают возможности работать с БД в полном объеме. Поэтому список фотографий для загрузки на форму необходимо брать из конкретной папки. Содержимое папки будет изменяться, поэтому не делайте «жесткой» привязки по полному пути к фото. Есть вероятность, что данная папка со временем будет иметь структуру подпапок. В этом случае поиск файлов необходимо делать во всех подпапках и формировать список загружаемых фото по результатам этого поиска. Вы разрабатываете модуль, в котором загрузка фото из папок на форме оформлена в виде слайдера- по 3 фото за 1 раз. Смена фото происходит по нажатию соответствующих кнопок. Макет формы представлен на рисунке 1. Алгоритм изменения изображений должен работать таким образом, чтобы учесть случай, когда количество фото не кратно 3. В этом случае,в последней группе из 2-х фото добавится самое первое фото в папке(сдвиг произойдет на 1 позицию) и т.д.

Скриншоты результатов работы программы

Graphical user interface, website

Description automatically generated

Листинги кода

Разметка XAML:

<Window x:Class="MEADDITIONAL.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:MEADDITIONAL"

mc:Ignorable="d"

Title="NBA Management System" Height="700" Width="800">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="10\*"/>

<RowDefinition Height="20\*"/>

<RowDefinition Height="15\*"/>

<RowDefinition Height="5\*"/>

</Grid.RowDefinitions>

<TextBlock Text="NBA Managment System" FontWeight="Bold" FontSize="24" Foreground="DarkSeaGreen" VerticalAlignment="Center" HorizontalAlignment="Center" Margin="20" FontFamily="Arial"/>

<TextBlock Grid.Row="1" Text="Welcome. To use this NBA Managment system, you can redirect to different pages according to your role by clicking the buttons below" FontWeight="Normal" FontSize="18" Foreground="Gray" VerticalAlignment="Top" HorizontalAlignment="Center" MaxWidth="400" TextAlignment="Center" FontFamily="Arial" TextWrapping="Wrap" Margin="30"/>

<Image Source="E:\MEADDITIONAL\logo.jpg" HorizontalAlignment="Left" Margin="10" Grid.Row="1"/>

<StackPanel Grid.Row="1" Orientation="Horizontal" HorizontalAlignment="Center" VerticalAlignment="Bottom" Margin="0,0,0,40" Background="LightGray">

<Button Background="DarkSeaGreen" Content="Visitor" Height="50" Width="150" Margin="20" FontFamily="Arial"/>

<Button Background="DarkSeaGreen" Content="Admin" Height="50" Width="150" Margin="20" FontFamily="Arial"/>

</StackPanel>

<StackPanel Grid.Row="2" Orientation="Horizontal" HorizontalAlignment="Center" Margin="15">

<Button Name="Left" Grid.Row="2" Background="Transparent" Content="←" Height="40" FontSize="20" Width="40" Margin="1" BorderThickness="0" HorizontalAlignment="Left" Click="RightButton\_Click"/>

<StackPanel Grid.Row="2" Orientation="Horizontal">

<Image x:Name="Image1" Width="200" Source="images/Group 1.png" Margin="10" Stretch="Uniform" />

<Image x:Name="Image2" Width="200" Source="images/Group 2.png" Margin="10" Stretch="Uniform"/>

<Image x:Name="Image3" Width="200" Source="images/Group 3.jpg" Margin="10" Stretch="Uniform"/>

</StackPanel>

<Button Name="Right" Grid.Row="2" Background="Transparent" BorderThickness="0" Content="🠒" Height="40" Width="40" Margin="1" FontSize="24" HorizontalAlignment="Right" Click="LeftButton\_Click"/>

</StackPanel>

<Rectangle Grid.Row="3" Fill="DarkSeaGreen"/>

<TextBlock Grid.Row="3" Text="The current season is 2016-2017, and the NBA has a history of 71 years" FontWeight="Normal" FontSize="18" Foreground="Black" VerticalAlignment="Center" HorizontalAlignment="Center" TextAlignment="Center" FontFamily="Arial" />

</Grid>

</Window>

Логика:

using System;

using System.IO;

using System.Reflection;

using System.Security.Policy;

using System.Threading;

using System.Windows;

using System.Windows.Media.Imaging;

namespace MEADDITIONAL

{

public partial class MainWindow : Window

{

string[] files = Directory.GetFiles(@"E:\MEADDITIONAL\images", "\*.jpg", SearchOption.AllDirectories);

public int selectedItem = 2;

public MainWindow()

{

InitializeComponent();

switch (files.Length)

{

case 1:

Image1.Source = new BitmapImage(new Uri(files[0], UriKind.RelativeOrAbsolute));

Image2.IsEnabled = false;

Image3.IsEnabled = false;

Left.IsEnabled = false;

Right.IsEnabled = false;

break;

case 2:

Image1.Source = new BitmapImage(new Uri(files[0], UriKind.RelativeOrAbsolute));

Image2.Source = new BitmapImage(new Uri(files[1], UriKind.RelativeOrAbsolute));

Image3.IsEnabled = false;

Left.IsEnabled = false;

Right.IsEnabled = false;

break;

case 3:

Image1.Source = new BitmapImage(new Uri(files[0], UriKind.RelativeOrAbsolute));

Image2.Source = new BitmapImage(new Uri(files[1], UriKind.RelativeOrAbsolute));

Image3.Source = new BitmapImage(new Uri(files[2], UriKind.RelativeOrAbsolute));

Left.IsEnabled = false;

Right.IsEnabled = false;

break;

default:

Image1.Source = new BitmapImage(new Uri(files[0], UriKind.RelativeOrAbsolute));

Image2.Source = new BitmapImage(new Uri(files[1], UriKind.RelativeOrAbsolute));

Image3.Source = new BitmapImage(new Uri(files[2], UriKind.RelativeOrAbsolute));

break;

}

}

private void SwapPhoto(bool direction)

{

if (direction)

{

if (selectedItem + 5 < files.Length)

{

selectedItem += 3;

Image1.Source = new BitmapImage(new Uri(files[selectedItem - 2], UriKind.RelativeOrAbsolute));

Image2.Source = new BitmapImage(new Uri(files[selectedItem - 1], UriKind.RelativeOrAbsolute));

Image3.Source = new BitmapImage(new Uri(files[selectedItem], UriKind.RelativeOrAbsolute));

}

else

{

selectedItem = 2;

Image1.Source = new BitmapImage(new Uri(files[0], UriKind.RelativeOrAbsolute));

Image2.Source = new BitmapImage(new Uri(files[1], UriKind.RelativeOrAbsolute));

Image3.Source = new BitmapImage(new Uri(files[2], UriKind.RelativeOrAbsolute));

}

}

else

{

if (selectedItem - 5 >= 0)

{

selectedItem -= 3;

Image1.Source = new BitmapImage(new Uri(files[selectedItem], UriKind.RelativeOrAbsolute));

Image2.Source = new BitmapImage(new Uri(files[selectedItem + 1], UriKind.RelativeOrAbsolute));

Image3.Source = new BitmapImage(new Uri(files[selectedItem + 2], UriKind.RelativeOrAbsolute));

}

else

{

selectedItem = files.Length - 4;

Image1.Source = new BitmapImage(new Uri(files[files.Length - 3], UriKind.RelativeOrAbsolute));

Image2.Source = new BitmapImage(new Uri(files[files.Length - 2], UriKind.RelativeOrAbsolute));

Image3.Source = new BitmapImage(new Uri(files[files.Length - 1], UriKind.RelativeOrAbsolute));

}

}

}

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

{

SwapPhoto(false);

}

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

{

SwapPhoto(true);

}

}

}

# Лабораторная работа №3

**Тема:** Создание собственных классов в C#.

**Цель работы:** получение практических навыков при создании и наследовании классов в C#.

**Text

Description automatically generated**

**10 вариант**

Скриншоты работы программы

Table

Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidence

Листинги кода:

Класс и его методы (Кв. Матрица)

using System;

using System.Collections.Generic;

using System.Data.Common;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace L03Console

{

public class SquareMatrix

{

private int size;

private List<List<double>> values;

public int Size

{

get { return size; }

set

{

if (value <= 0)

{

throw new Exception("ERROR: Size should be greater than 0.");

}

size = value;

}

}

public List<List<double>> Values

{

get { return values; }

set

{

if (value.Count != Size || value[0].Count != Size)

{

throw new Exception("ERROR: Invalid input");

}

values = value;

}

}

// конструкторы

public SquareMatrix(int size, List<List<double>> values) { Size = size; Values = values; }

public SquareMatrix(int size)

{

Size = size;

var values = new List<List<double>>();

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

{

values.Add(new List<double>());

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

{

var rand = new Random();

values.Last().Add(Math.Round(rand.NextDouble() \* 10, 2));

}

}

Values = values;

}

// методы

public double MatrixSum()

{

int indexX = 0;

int indexY = 0;

double sum = 0;

while (indexX != values.Count)

{

while (indexY != values[indexX].Count)

{

sum += values[indexX].Sum();

indexY++;

}

indexX++;

}

return Math.Round(sum, 2);

}

public SquareMatrix Plus(SquareMatrix operand)

{

SquareMatrix initial = new(operand.Size);

if (this.Size != operand.Size) throw new Exception("ERROR: Incorrect size of operands");

for (int i = 0; i < this.Size; i++)

{

for (int j = 0; j < this.Size; j++)

initial.Values[i][j] = Math.Round(this.Values[i][j] + operand.Values[i][j], 2);

}

return initial;

}

public SquareMatrix Minus(SquareMatrix operand)

{

SquareMatrix initial = new(operand.Size);

if (this.Size != operand.Size) throw new Exception("ERROR: Incorrect size of operands");

for (int i = 0; i < this.Size; i++)

{

for (int j = 0; j < this.Size; j++)

initial.Values[i][j] = Math.Round(this.Values[i][j] - operand.Values[i][j], 2);

}

return initial;

}

public SquareMatrix MultiplyConstant(int constant)

{

SquareMatrix initial = new(this.Size);

for (int i = 0; i < this.Size; i++)

{

for (int j = 0; j < this.Size; j++)

initial.Values[i][j] = Math.Round(this.Values[i][j] \* constant, 2);

}

return initial;

}

public SquareMatrix Multiply(SquareMatrix operand)

{

if (operand.Size != this.Size) throw new Exception("ERROR: Incorrect size of operands.");

var initial = new SquareMatrix(this.Size);

for (int i = 0; i < this.Size; i++)

{

for (int j = 0; j < this.Size; j++)

{

double val = 0;

for (int z = 0; z < this.Size; z++)

{

val += Math.Round(this.Values[i][z] \* operand.Values[z][j], 2);

}

initial.Values[i][j] = Math.Round(val, 2);

}

}

return initial;

}

public static SquareMatrix operator +(SquareMatrix matrix1, SquareMatrix matrix2)

{

if (matrix1.Size != matrix2.Size) throw new Exception("ERROR: Invalid sizes");

SquareMatrix initial = new(matrix2.Size);

for (int i = 0; i < matrix1.Size; i++)

{

for (int j = 0; j < matrix1.Size; j++)

initial.Values[i][j] = Math.Round(matrix1.Values[i][j] + matrix2.Values[i][j], 2);

}

return initial;

}

public static SquareMatrix operator -(SquareMatrix matrix1, SquareMatrix matrix2)

{

if (matrix1.Size != matrix2.Size) throw new Exception("ERROR: Invalid sizes");

SquareMatrix initial = new(matrix2.Size);

for (int i = 0; i < matrix1.Size; i++)

{

for (int j = 0; j < matrix1.Size; j++)

initial.Values[i][j] = Math.Round(matrix1.Values[i][j] - matrix2.Values[i][j], 2);

}

return initial;

}

public static SquareMatrix operator \*(SquareMatrix matrix1, int constant)

{

SquareMatrix initial = new(matrix1.Size);

for (int i = 0; i < matrix1.Size; i++)

{

for (int j = 0; j < matrix1.Size; j++)

initial.Values[i][j] = Math.Round(matrix1.Values[i][j] \* constant, 2);

}

return initial;

}

public static SquareMatrix operator \*(SquareMatrix matrix1, SquareMatrix matrix2)

{

if (matrix2.Size != matrix2.Size) throw new Exception("ERROR: Incorrect size of operands.");

var initial = new SquareMatrix(matrix1.Size);

for (int i = 0; i < matrix1.Size; i++)

{

for (int j = 0; j < matrix1.Size; j++)

{

double val = 0;

for (int z = 0; z < matrix1.Size; z++)

{

val += Math.Round(matrix1.Values[i][z] \* matrix2.Values[z][j], 2);

}

initial.Values[i][j] = Math.Round(val, 2);

}

}

return initial;

}

public override string ToString()

{

string matrixStr = "\n";

matrixStr += new StringBuilder().Insert(0, "--", Size\*6).ToString();

matrixStr += "\n";

foreach (var list in values)

{

matrixStr += '\t' + String.Join('\t', list) + '\n';

}

matrixStr += new StringBuilder().Insert(0, "--", Size\*6).ToString();

return matrixStr;

}

}

}

Основная программа

using L03Console;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

var test = new SquareMatrix(3);

var testOperand = new SquareMatrix(3);

Console.WriteLine("Sum of matrixes:");

Console.WriteLine(test.ToString() + "\n\n\t\t+");

Console.WriteLine(testOperand.ToString());

Console.WriteLine("\n\t\t=");

Console.WriteLine(test.Plus(testOperand).ToString());

Console.WriteLine("Matrix sum:"); Console.WriteLine(test.MatrixSum() + "\n");

Console.WriteLine("\n\nConstant Multiply Test\n----------\n");

Console.WriteLine(test.MultiplyConstant(3));

Console.WriteLine("\n\nMultiply Test\n----------\n");

Console.WriteLine(test.Multiply(testOperand));

Console.WriteLine();

Console.WriteLine((test + testOperand).ToString());

Console.WriteLine("\n\nMultiply Constant Test 2\n----------\n");

Console.WriteLine((test\*2).ToString());

Console.WriteLine();

Console.WriteLine("\n\nMultiply Test 2\n----------\n");

Console.WriteLine((test \* testOperand).ToString());