

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 Разработка программных модулей

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. №

021к

В.Д. Панков

подпись, дата

инициалы, фамилия

Санкт-Петербург 2023

СОДЕРЖАНИЕ

1 Лабораторная работа № 1	2
2 Дополнительные задания на работу с файловой системой.....	10
3 Лабораторная работа №3	26
4 Лабораторная работа № 4	51
4.1 Часть 1.....	51
4.2 Часть 2.....	56
5 Лабораторная работа №5	77

1 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.

Примечание. При выполнении задания необходимо работать с типом DateTime.

DateTime.Now – возвращает текущее время;

Convert.ToDateTime – преобразование строки в тип DateTime.

Самостоятельно необходимо разобраться как работать с секундами.

Решение:

Разметка:

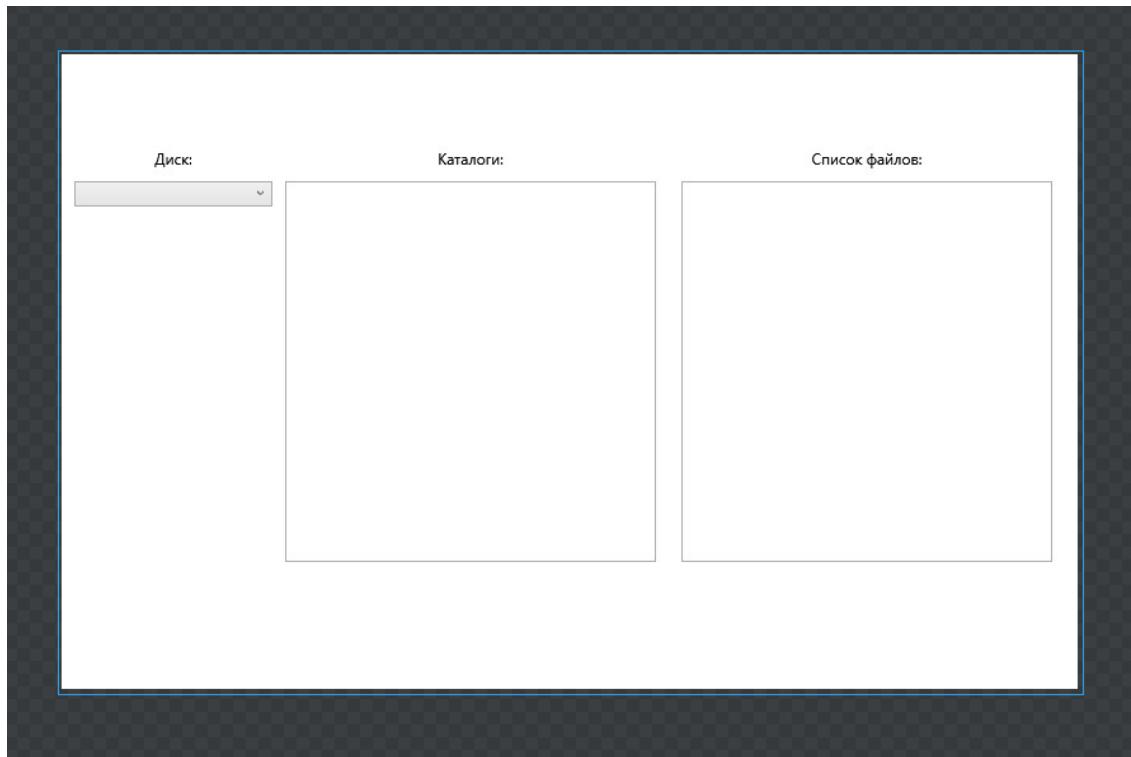
```
<Window x:Class="WorkWithFiles.MainWindow"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d"
    Title="MainWindow" Height="500" Width="800"
    MinHeight="500" MinWidth="700">
    <Grid Margin="10, 0">
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="*" />
            <ColumnDefinition Width="2*" />
            <ColumnDefinition Width="2*" />
        </Grid.ColumnDefinitions>
        <Grid.RowDefinitions>
```

```

        <RowDefinition Height="*" />
        <RowDefinition Height="3*" />
        <RowDefinition Height="*" />
    </Grid.RowDefinitions>
    <TextBlock TextAlignment="Center" VerticalAlignment="Bottom"
        ↳ Margin="10">Диск:</TextBlock>
    <StackPanel Grid.Row="1" Grid.Column="0" Orientation="Vertical">
        <ComboBox SelectionChanged="HardDrives_OnSelected"
            ↳ Name="HardDrivesComboBox" Height="20"
                VerticalAlignment="Top" />
        <TextBlock Name="HardDriveInformation" TextWrapping="Wrap" />
    </StackPanel>
    <TextBlock Grid.Row="0" Grid.Column="1" TextAlignment="Center"
        ↳ VerticalAlignment="Bottom" Margin="10">Каталоги:</TextBlock>
    <TextBlock Margin="10" Grid.Row="2" Grid.Column="1"
        ↳ Name="FolderInformation" DockPanel.Dock="Bottom"
            TextWrapping="Wrap" />
    <ListBox Grid.Row="1" Grid.Column="1" VerticalAlignment="Stretch"
        SelectionChanged="FoldersListBox_OnSelectionChanged"
        ↳ Name="FoldersListBox"
        Margin="10, 0" />

    <TextBlock Grid.Row="0" Grid.Column="2" TextAlignment="Center"
        ↳ VerticalAlignment="Bottom" Margin="10">Список файлов:</TextBlock>
    <ListBox Grid.Column="2" Name="FilesListBox" Grid.Row="1"
        ↳SelectionMode="Single"
            SelectionChanged="FilesListBox_OnSelectionChanged"
            Margin="10, 0" />
</Grid>
</Window>

```



Код приложения:

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.IO;
using System.Linq;
using System.Windows;
using System.Windows.Controls;

namespace WorkWithFiles;

/// <summary>
/// Interaction logic for MainWindow.xaml
/// </summary>
public partial class MainWindow : Window
{
    public MainWindow()
    {
        // Создаём и очищаем файл
        var streamWriter = new StreamWriter("files.txt");
```

```

streamWriter.Write("");
streamWriter.Close();
InitializeComponent();
// Заполняем поле дисков
foreach (var drive in DriveInfo.GetDrives())
    HardDrivesComboBox.Items.Add(drive.Name);
HardDrivesComboBox.SelectedIndex = 0;
}

private void HardDrives_OnSelected(object sender, RoutedEventArgs e)
{
    FoldersListBox.Items.Clear();
    FilesListBox.Items.Clear();
    var hardDrive = HardDrivesComboBox.SelectedItem.ToString()!;
    foreach (var directory in Directory.GetDirectories(hardDrive))
        FoldersListBox.Items.Add(directory);
    // Получаем информацию о диске
    var driveInfo = DriveInfo.GetDrives().Where(info => info.Name == hardDrive);
    HardDriveInformation.Text = @$"Объем диска: {driveInfo.TotalSize / (1024 * 1024)} ГБ";
    Свободное пространство: {driveInfo.TotalFreeSpace / (1024 * 1024)} MB";
}

private void FoldersListBox_OnSelectionChanged(object sender, SelectionChangedEventArgs e)
{
    FilesListBox.Items.Clear();
    FolderInformation.Text = "";
    try
    {
        var path = FoldersListBox.SelectedItem.ToString()!;
        var DirectoryInfo = new DirectoryInfo(path);
        FolderInformation.Text = @$"Полное название каталога: {directoryInfo.FullName}";
        Время создания каталога: {directoryInfo.CreationTime};
        Корневой каталог: {directoryInfo.Parent?.Name ?? string.Empty}";
    }
}

```

```
foreach (var file in Directory.GetFiles(path))
    FilesListBox.Items.Add(file);
}
catch (NullReferenceException)
{
/* При очистке поле Selection сбрасывается и нас выкидывает сюда, так */
}
catch (UnauthorizedAccessException)
{
    MessageBox.Show("Доступ запрещён к этой папке.");
}
}

/* На открытие любого файла или при закрытии приложения
производим проверку все ли записанные файлы, открывались 10 секунд назад
*/
private void fileOpened(string? path = null)
{
    HashSet<(string, DateTime)> notDelete = new();
    if (path != null)
        notDelete.Add((path, DateTime.Now));
    var streamReader = new StreamReader("files.txt");
    while (!streamReader.EndOfStream)
    {
        var line = streamReader.ReadLine()!;
        if (line.Trim() != "")
        {
            var runnedTime = DateTime.Parse(line.Split(',') [1]);
            if (DateTime.Now - runnedTime < TimeSpan.FromSeconds(10))
                notDelete.Add((line.Split(',') [0], runnedTime));
        }
    }
}

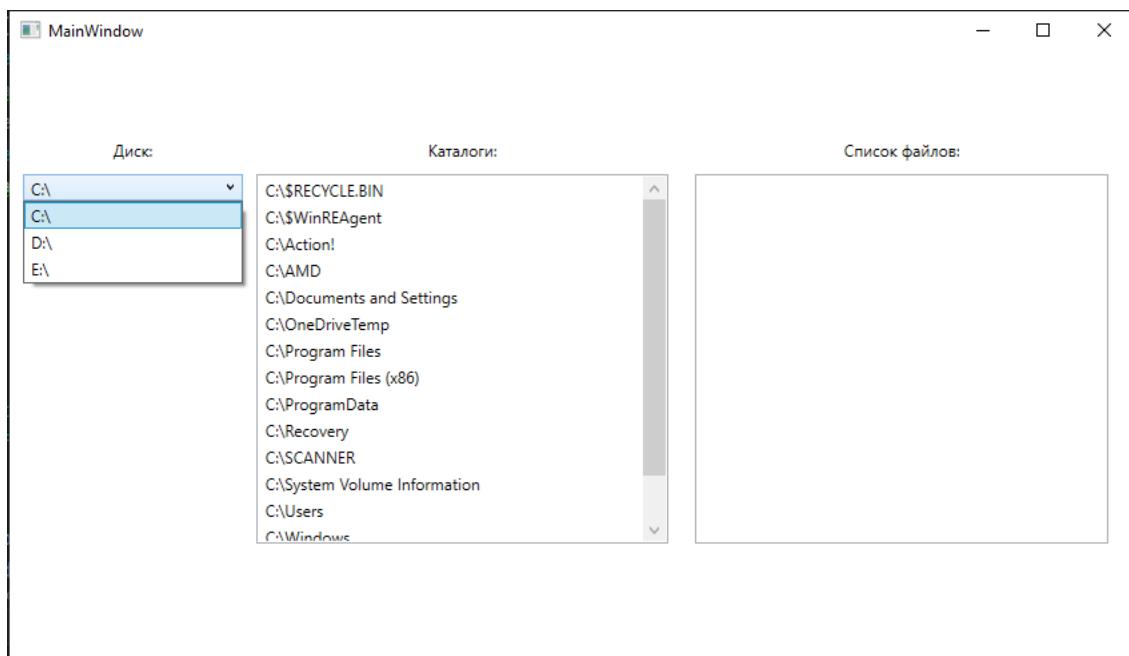
streamReader.Close();
```

```
var streamWriter = new StreamWriter("files.txt");
foreach (var pair in notDelete)
    streamWriter.WriteLine($"{pair.Item1},{pair.Item2}");
streamWriter.Close();
}

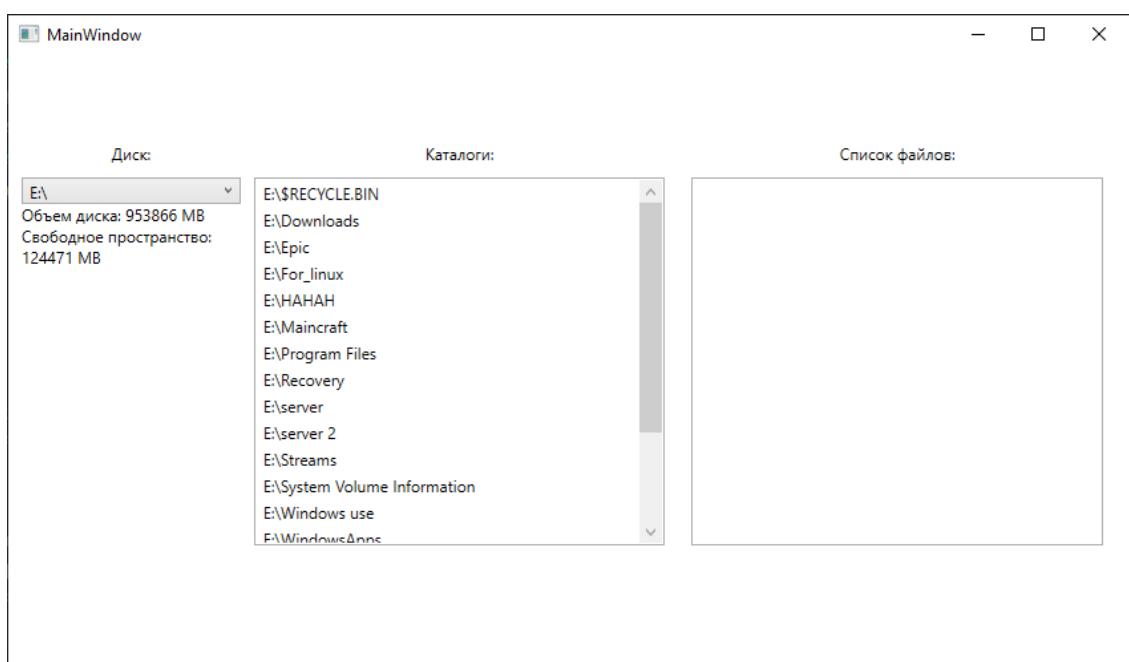
protected override void OnClosed(EventArgs e)
{
    fileOpened();
    base.OnClosed(e);
}

private void FilesListBox_OnSelectionChanged(object sender, SelectionC
{
    try
    {
        var path = FilesListBox.SelectedItem.ToString()!;
        new Process
        {
            StartInfo = new ProcessStartInfo(path)
            {
                UseShellExecute = true
            }
            .Start();
        fileOpened(path);
        FilesListBox.SelectedItem = null;
    }
    catch (NullReferenceException)
    {
    }
}
}
```

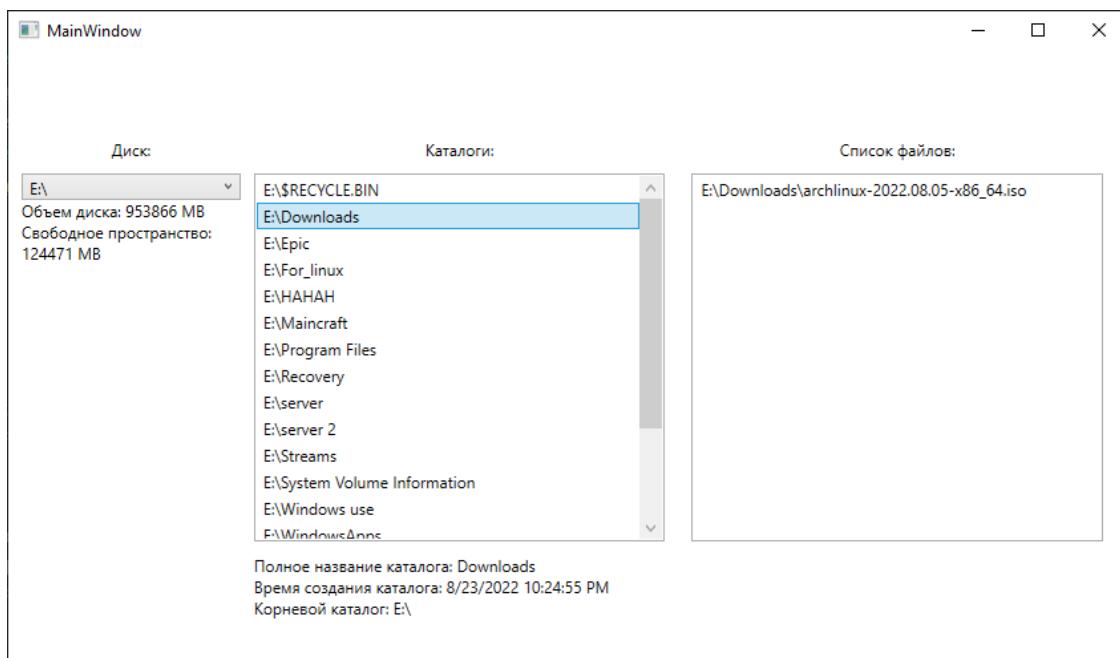
Демонстрация работы приложения:



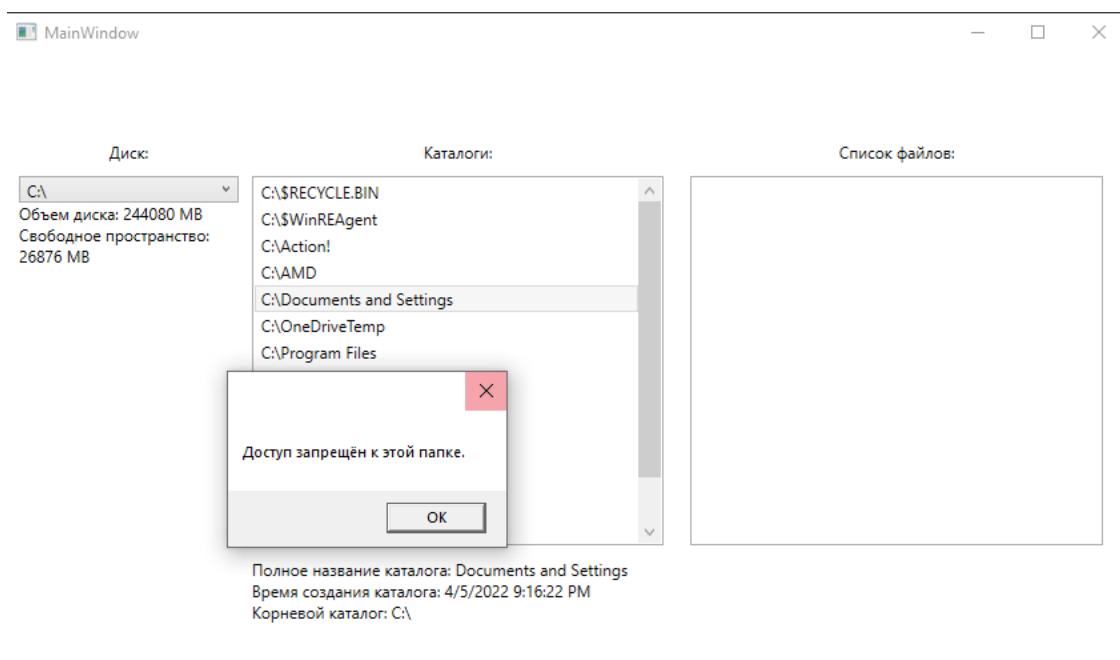
Выбран диск:



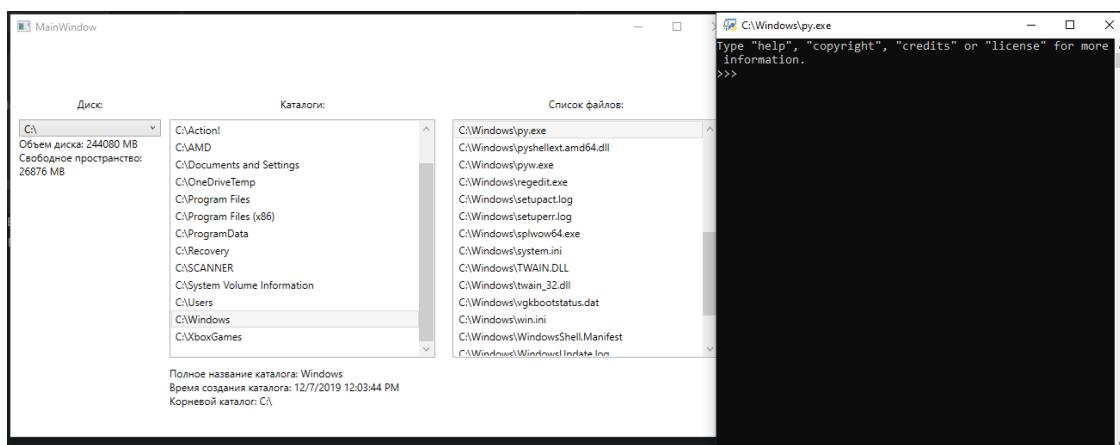
Выбрана папка:



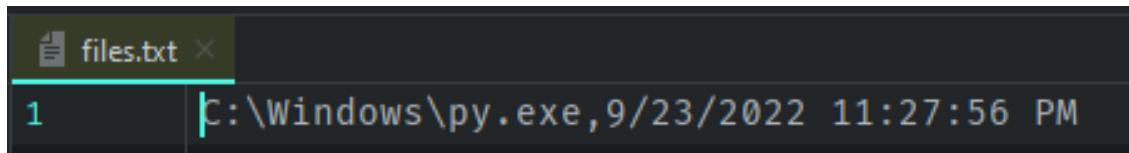
Обработка ошибки с доступом:



Открытие файла(ru . exe)



Запись открытия данного файла:



```
files.txt
1 C:\Windows\py.exe, 9/23/2022 11:27:56 PM
```

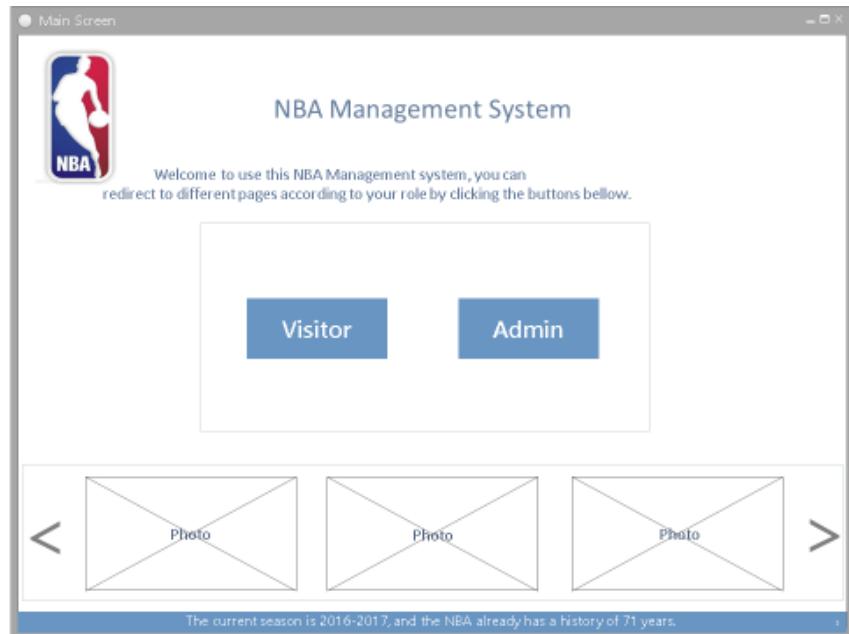
Итог работы:

Получил практические навыки при работе с пространством имен System.IO, а также создал приложение "Микропроводник".

2 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.

This is the main screen of the application. All users will see this when the application is started.



Для выполнения задания используйте папки с ресурсами – папку logo и папку Pictures.

Код:

Разметка:

```
<Window x:Class="NbaApp.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"
        mc:Ignorable="d"
        Title="MainWindow" Height="450" Width="800"
        MinHeight="500" MinWidth="800">
    <Grid>
        <Grid.ColumnDefinitions>
            <ColumnDefinition />
            <ColumnDefinition />
            <ColumnDefinition />
            <ColumnDefinition />
        </Grid.ColumnDefinitions>
        <Grid.RowDefinitions>
            <RowDefinition />
            <RowDefinition />
            <RowDefinition />
            <RowDefinition />
            <RowDefinition Height="20" />
        </Grid.RowDefinitions>

        <Image HorizontalAlignment="Left" VerticalAlignment="Top" Margin="10"
               Source="res/logo/logo.jpg" />
        <TextBlock FontSize="22" FontWeight="Medium" Grid.ColumnSpan="2"
                  HorizontalAlignment="Center" />
```

```

        VerticalAlignment="Center" Grid.Column="1">
    NBA Management System
</TextBlock>
<TextBlock Grid.Row="1" Grid.Column="1" Grid.ColumnSpan="2"
    ↳ FontSize="18" TextWrapping="Wrap"
        VerticalAlignment="Top" HorizontalAlignment="Center"
        ↳ TextAlignment="Center">
    Welcome to use this NBA Management system, you can redirect to
    ↳ different pages according to your role by clicking the
    ↳ buttons bellow
</TextBlock>
<Button Grid.Column="1" VerticalAlignment="Top" Margin="30"
    ↳ Foreground="White" Grid.Row="2">
    <TextBlock Margin="10">
        <Run Foreground="White">
            Visitor
        </Run>
    </TextBlock>
</Button>
<Button Grid.Column="2" Margin="30" VerticalAlignment="Top"
    ↳ Foreground="White" Grid.Row="2">
    <TextBlock Margin="10">
        <Run Foreground="White">
            Admin
        </Run>
    </TextBlock>
</Button>

<Grid Margin="10" Grid.Row="3" Grid.ColumnSpan="4">
    <Grid.ColumnDefinitions>
        <ColumnDefinition Width="*" />
        <ColumnDefinition Width="7*" />
        <ColumnDefinition Width="*" />
    </Grid.ColumnDefinitions>
    <Grid Grid.Column="1">
        <Grid.ColumnDefinitions>
            <ColumnDefinition />
            <ColumnDefinition />
            <ColumnDefinition />
        </Grid.ColumnDefinitions>
        <Image Margin="10, 0" Source="{Binding Img1}" />
        <Image Margin="10, 0" Grid.Column="1" Source="{Binding
            ↳ Img2}" />
        <Image Margin="10, 0" Grid.Column="2" Source="{Binding
            ↳ Img3}" />
    </Grid>

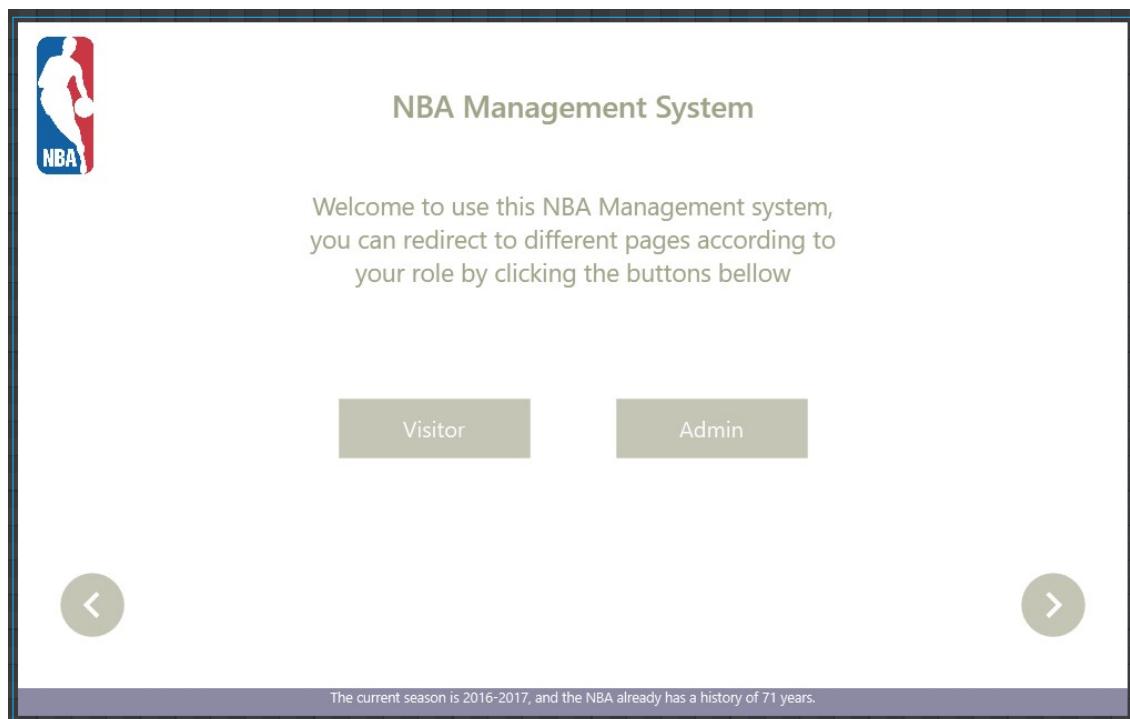
```

</Grid>

```

<Button Click="Left" Width="48" Height="48">
    <Button.Resources>
        <Style TargetType="{x:Type Border}">
            <Setter Property="CornerRadius"
                Value="123" />
        </Style>
    </Button.Resources>
    <Image Source="res/ui/left.png" Margin="3" />
</Button>
<Button Click="Right" Width="48" Height="48" Grid.Column="3">
    <Button.Resources>
        <Style TargetType="{x:Type Border}">
            <Setter Property="CornerRadius"
                Value="123" />
        </Style>
    </Button.Resources>
    <Image Source="res/ui/right.png" Margin="3" />
</Button>
</Grid>
<TextBlock Background="#8b89a4" Foreground="White"
    HorizontalAlignment="Stretch" TextAlignment="Center"
    Grid.Row="4" Grid.ColumnSpan="4" FontSize="10">
    The current season is 2016-2017, and the NBA already has a
    history of 71 years.
</TextBlock>
</Grid>
</Window>

```



MainWindow.cs

```

using System;
using System.Collections.Generic;
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;

namespace NbaApp;

/// <summary>
/// Interaction logic for MainWindow.xaml
/// </summary>
public partial class MainWindow : Window
{
    private Images _images = new();

    public MainWindow()
    {
        DataContext = _images;
        InitializeComponent();
    }

    private void Right(object sender, RoutedEventArgs e)
    {
        _images.Right();
    }

    private void Left(object sender, RoutedEventArgs e)
    {
        _images.Left();
    }
}

```

Images.cs

```

using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;

namespace NbaApp;

```

```

public class Images : ViewModel
{
    // Путь до картинок
    private const string PATH = @"C:\Users\user\Desktop\NbaApp\NbaApp\res\";
    private string[] _showedImages;

    private int _index = 0;

    // Получаем все картинки из подпапок, а также их фильтруем по *.jpg
    private List<string> _images = Directory.EnumerateFiles(PATH, "*.jpg",
    ↵ SearchOption.AllDirectories).ToList();

    // Параметры для binding
    public string Img1
    {
        get => _showedImages[0];
        set => _showedImages[0] = value;
    }

    public string Img2
    {
        get => _showedImages[1];
        set => _showedImages[1] = value;
    }

    public string Img3
    {
        get => _showedImages[2];
        set => _showedImages[2] = value;
    }

    public Images()
    {
        ShowImages();
    }

    // Функция отображения картинок по индексу
    private void ShowImages()
    {
        if (_images.Count == 0)
            return;

        // Если у нас одна картинка просто зацикливаем одну картинку и
        ↵ наливаем на индексы
        if (_images.Count == 1)
            _showedImages = new[] { _images[0], _images[0], _images[0] };
        else if (_index + 3 >= _images.Count)
    }

```

```

{
    var list_ = _images.GetRange(_index, _images.Count - _index);
    list_.AddRange(_images.GetRange(0, 3 - (_images.Count -
→ _index)));
    _showedImages = list_.ToArray();
}
else
{
    _showedImages = _images.GetRange(_index, 3).ToArray();
}

// Уведомление WPF о изменении в этих Properties
NotifyPropertyChanged("Img1");
NotifyPropertyChanged("Img2");
NotifyPropertyChanged("Img3");
}

// Высчитываем новый индекс при листании вправо и меняем картинки
public void Right()
{
    if (_images.Count == 0)
        return;

    // Если индекс превышает допустимое значение, получаем новый циклический
→ индекс, иначе просто листаем на 3
    if (_index + 3 >= _images.Count)
        _index = 3 - (_images.Count - _index);
    else
        _index += 3;
    /* Существует ошибка если индекс меньше 3, то мы не получаем новый
→ корректно(вычитание из тройки),
это исправляет ошибку*/
    if (_images.Count < 3 && _index == _images.Count)
        _index = 0;
    ShowImages();
}

// Высчитываем новый индекс при листании влево и меняем картинки
public void Left()
{
    if (_images.Count == 0)
        return;
    if (_index - 3 <= 0)
        _index = Math.Abs(_images.Count + (_index - 3));
    else
        _index -= 3;
    ShowImages();
}

```

```
}
```

ViewModel.cs

```
using System.ComponentModel;

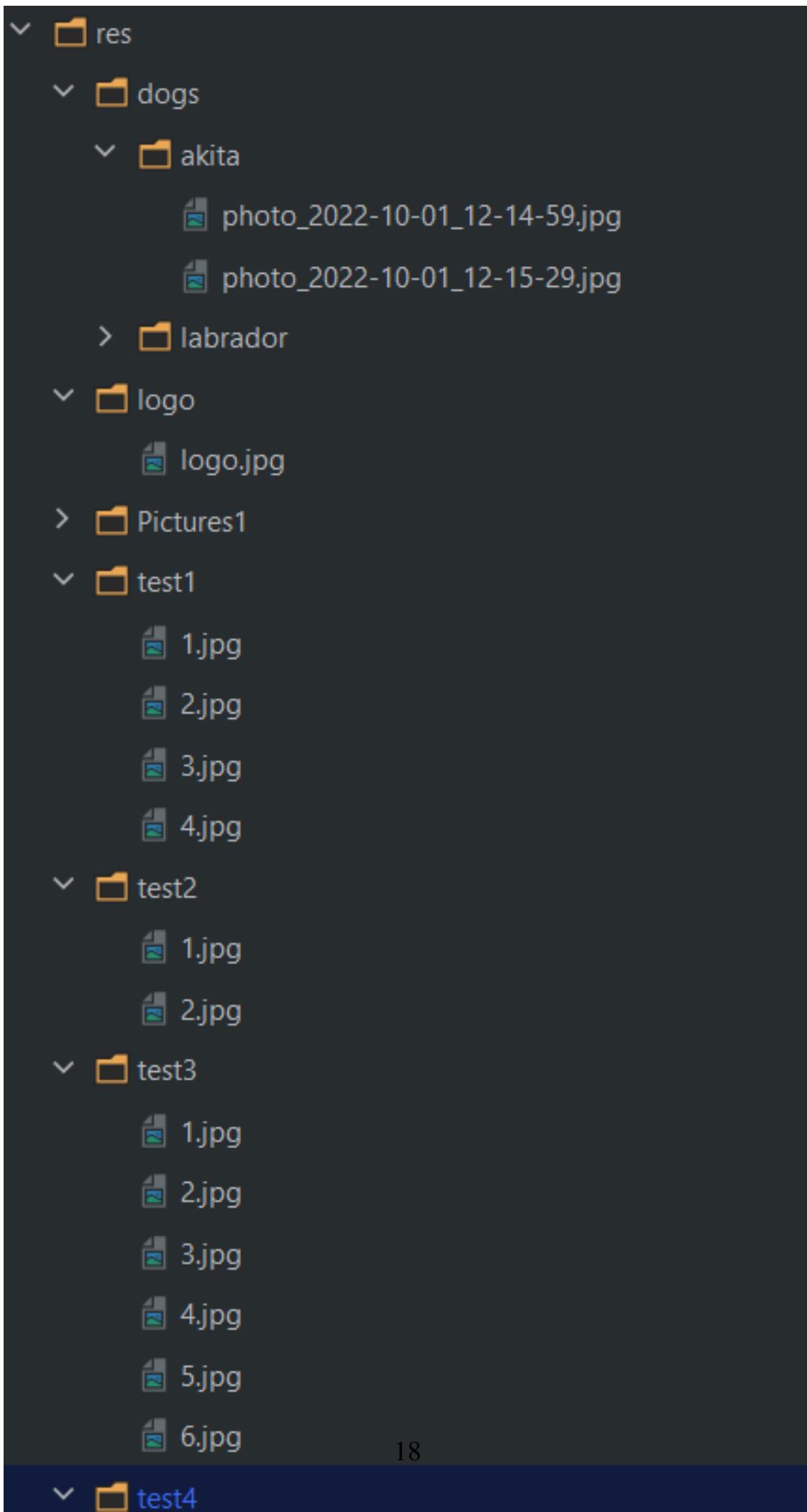
namespace NbaApp;

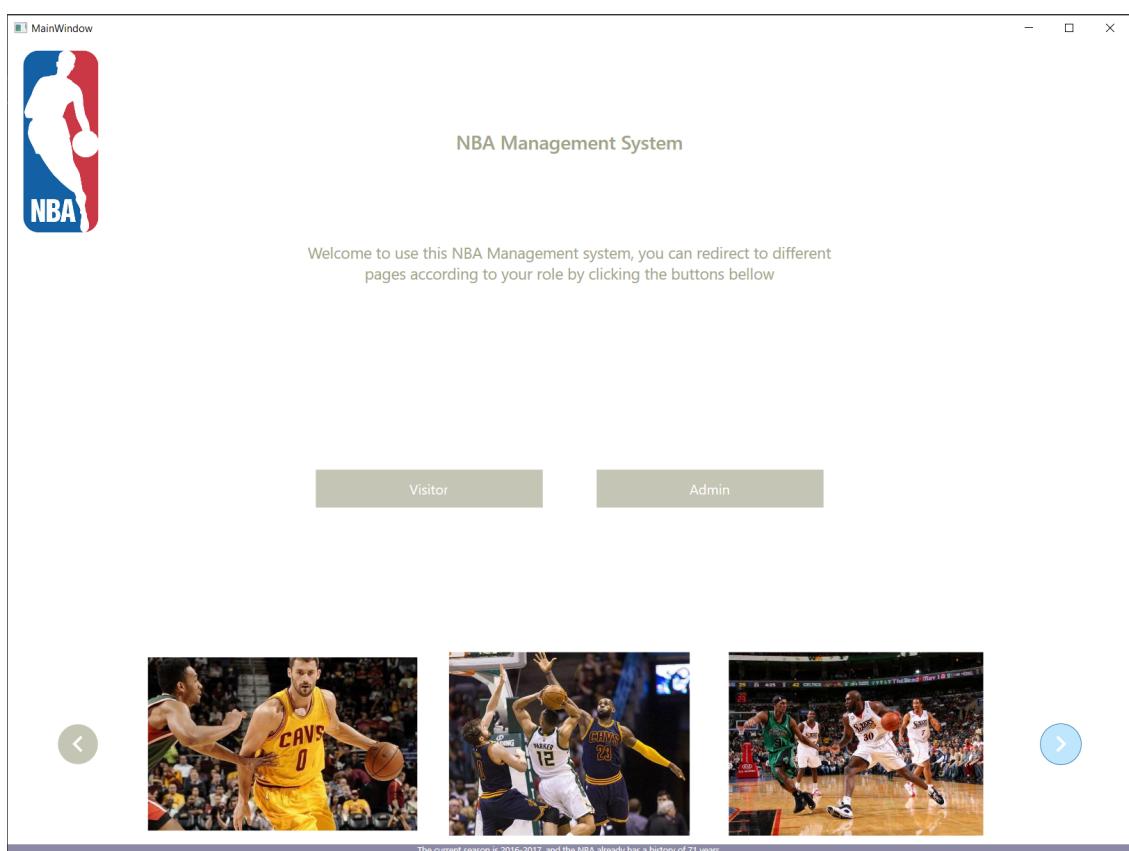
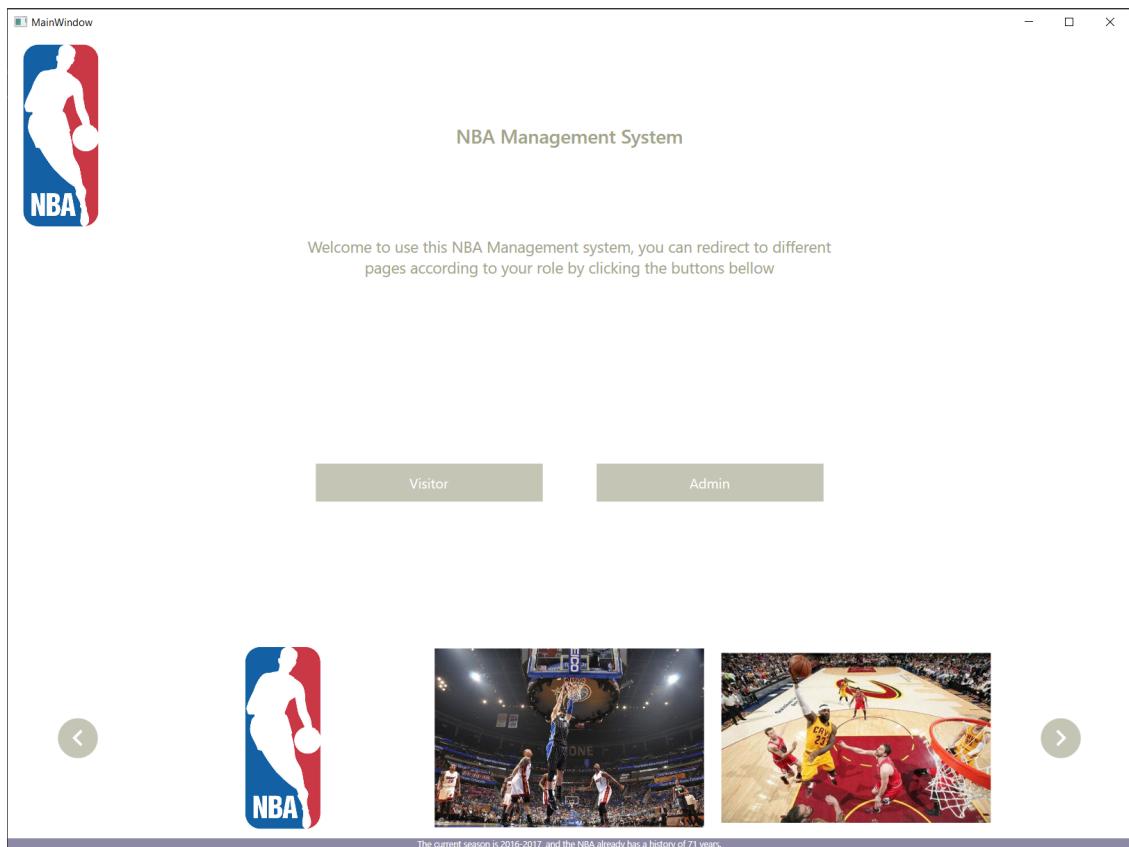
// Крупой класс для уведомления WPF, что Property изменилось
public abstract class ViewModel : INotifyPropertyChanged
{
    public event PropertyChangedEventHandler? PropertyChanged;

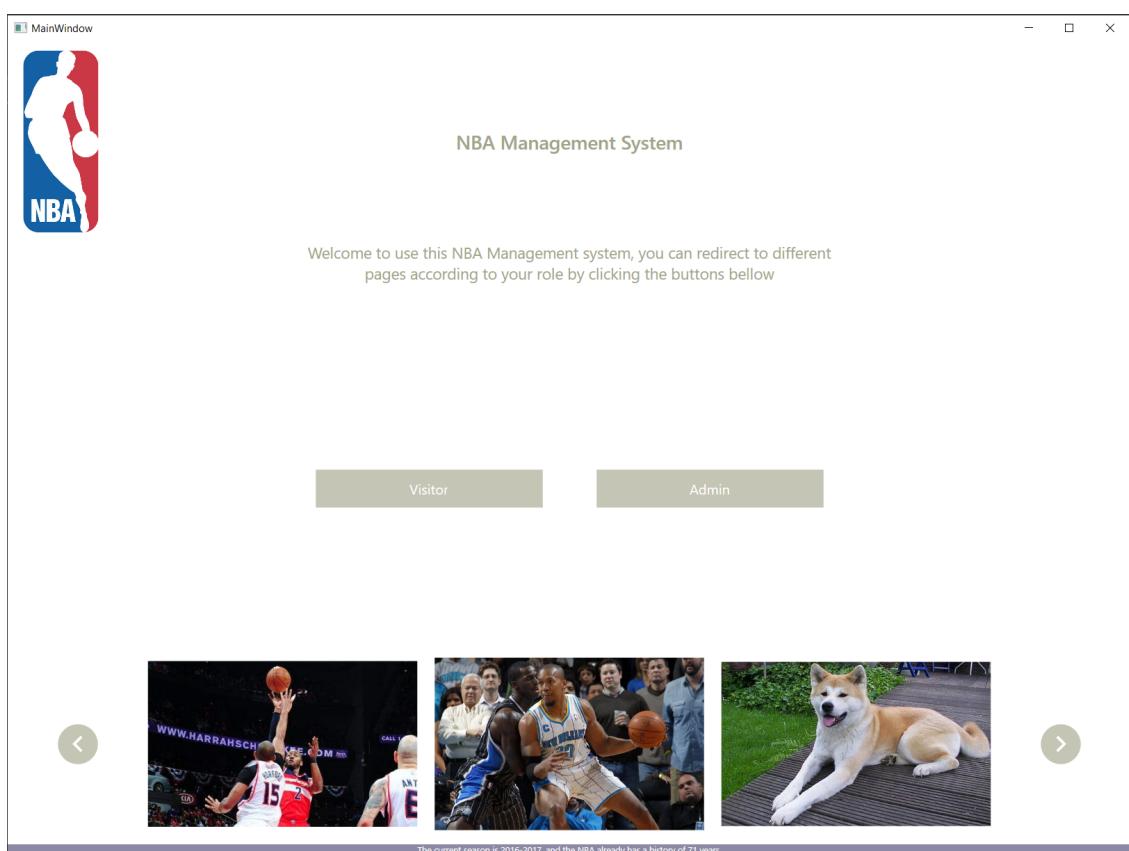
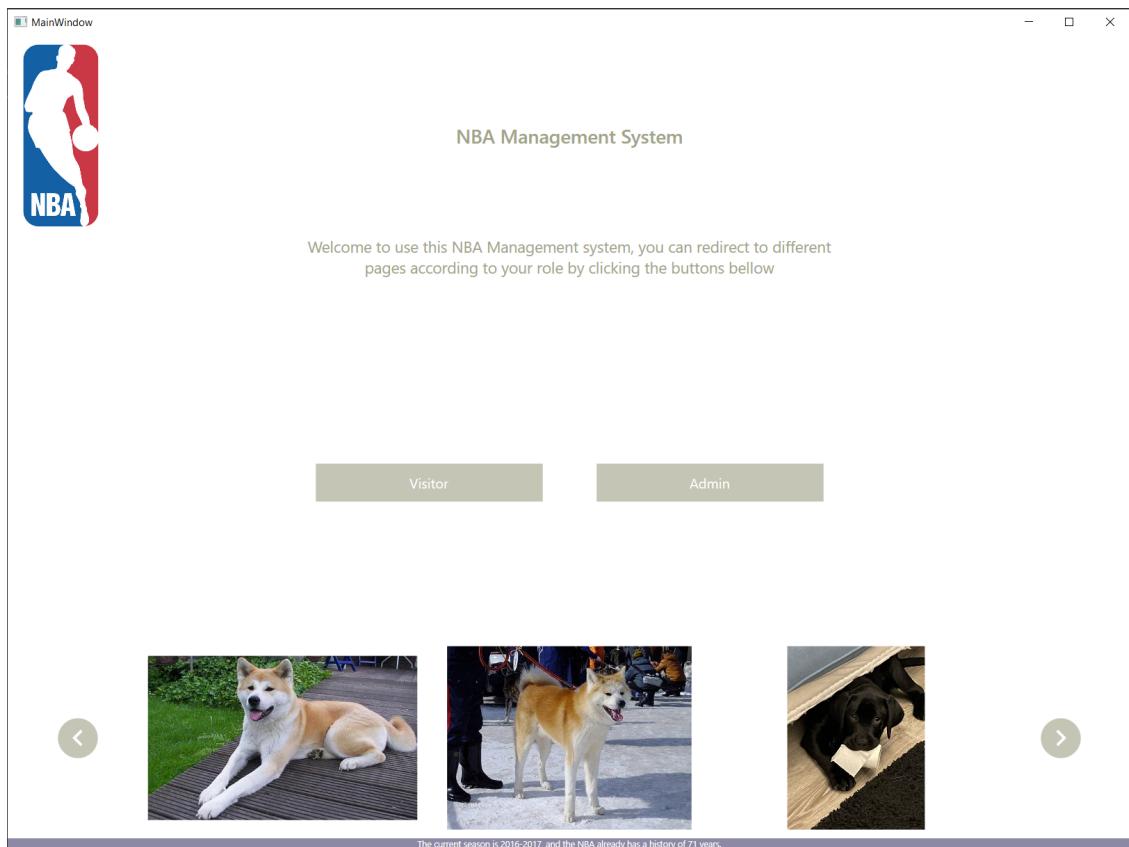
    protected void NotifyPropertyChanged(string propertyName)
    {
        if (PropertyChanged != null)
            PropertyChanged(this, new
                PropertyChangedEventArgs(propertyName));
    }
}
```

Демонстрация работы приложения:

Картинки используемые в приложении:







Задание 2.(фрагмент модуля задания по компетенции Программные решения для бизнеса(2017 год), overdrive на 30 минут).

К нам обратился заказчик со срочной задачей. С последнего чемпионата WorldSkills остался большой массив фотографий. Часть фотографий лежит общей кучей, часть рассортирована по папкам. Помимо этого, заказчик до обращения к нам привлекал сотрудников к ручной сортировке фотографий. Это была ручная сортировка. Люди вручную создавали папки и переносили или копировали туда фотографии. В итоге часть фотографий оказалась в нескольких экземплярах в различных папках и подпапках.

Заказчик хочет, чтобы вы написали простое приложение, которое позволит выполнить следующие задачи:

- 1) Осуществить поиск и удаление дублирующих файлов. Сравнение файлов должно происходить сразу по двум параметрам:
 - По наименованию файла;
 - По размеру файла.
- 2) Производить сортировку файлов путем создания папок и переноса туда соответствующих фотографий. Папка, представляет собой период времени со следующими возможными шагами: День, неделя, месяц. Период должен выбираться пользователем и после этого все папки должны создаваться соответствующим образом. По выбранному алгоритму так же должны переноситься фотографии в соответствующие папки. Фотографии переносятся по дате создания. Фотографии переносятся с выбранной папки, включая все подпапки. Не должны создаваться пустые папки без наполнения.

Код:

```
using System.Globalization;
```

```
Console.WriteLine("Укажите путь до папки с фото(По умолчанию - данная папка)");
var path = Console.ReadLine() ?? ".";
path = path.Trim() == "" ? "." : path;
Dictionary<string, string>, string> filesMap = new Dictionary<(string, string)>();
Console.WriteLine("Дубликаты: ");
foreach (var filePath in Directory.EnumerateFiles(path, "*.*", SearchOption.AllDirectories))
{
    var fileInfo = new FileInfo(filePath);
    if (filesMap.ContainsKey((fileInfo.Name, fileInfo.CreationTime.ToString("dd/MM/yyyy")))) {
        Console.WriteLine($"\"{fileInfo.FullName} {filesMap[(fileInfo.Name, fileInfo.CreationTime.ToString("dd/MM/yyyy"))]}\"");
    } else {
        filesMap.Add((fileInfo.Name, fileInfo.CreationTime.ToString("dd/MM/yyyy")));
    }
}
```

idiot:

```

Console.WriteLine("Выберите период (d - day, w - week, m - month)");
var period = Console.ReadKey();
Dictionary<string, List<string>> filesSorted = new Dictionary<string, List<string>>();

foreach (var pair in filesMap.Keys)
{
    var createdTime = DateTime.Parse(pair.Item2);
    try
    {
        string key = period.Key switch
        {
            ConsoleKey.D => $"{createdTime.DayOfYear} {createdTime.Year}",
            ConsoleKey.W => $"{createdTime.DayOfYear / 7} {createdTime.Year}",
            ConsoleKey.M => $"{createdTime.Month} {createdTime.Year}",
            _ => throw new ArgumentOutOfRangeException()
        };
        if (filesSorted.ContainsKey(key))
        {
            filesSorted[key].Add(filesMap[pair]);
        }
        else
            filesSorted[key] = new List<string>() {filesMap[pair]};
    }
    catch (ArgumentOutOfRangeException e)
    {
        goto idiot;
    }
}

Directory.SetCurrentDirectory(path);
Directory.CreateDirectory("sorted");
Directory.SetCurrentDirectory("sorted");
foreach (var dateString in filesSorted.Keys)
{
    ...
}

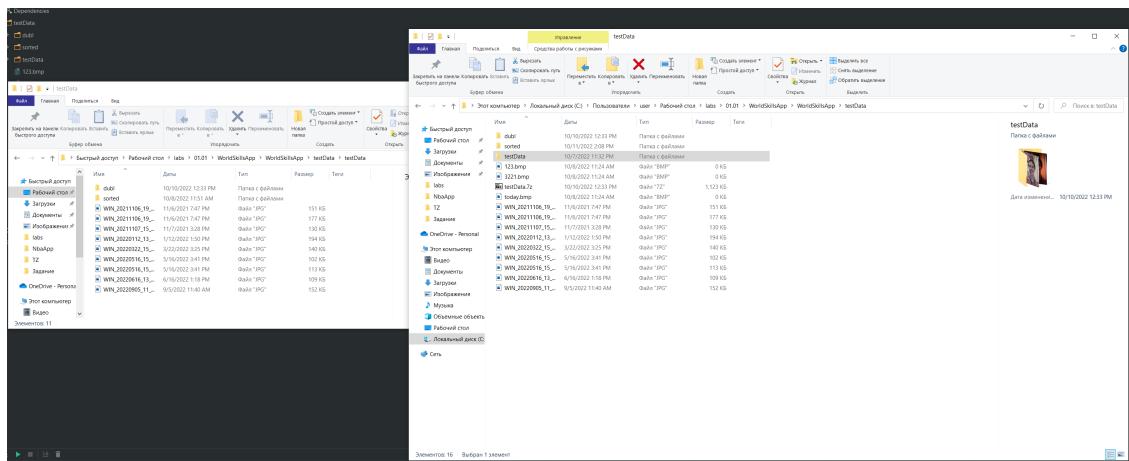
```

```

var ints = dateString.Split().Select(s => int.Parse(s)).ToArray();
(int what, int year) = (ints[0], ints[1]);
string folderName = period.Key switch
{
    ConsoleKey.D => $"{new DateTime(year, 1, 1).AddDays(what - 1).ToString("dd/MM/yyyy")}";
    ConsoleKey.W => new DateTime(year, 1, 1).AddDays(what * 7).ToString("dd/MM/yyyy");
    default => new DateTime(year, 1, 1).AddDays(what * 7 + 7).ToString("dd/MM/yyyy");
};
Console.WriteLine(folderName);
Directory.CreateDirectory(folderName);
foreach (var filePath in filesSorted[dateString])
{
    int fileRepeat = 0;
    again:
    try
    {
        var newFileName = new FileInfo(filePath).Name.Split('.')[0] + (fileRepeat > 0 ? $"-{fileRepeat}" : "") + new FileInfo(filePath).Name.Split('.')[1];
        File.Copy(filePath, folderName + "\\\" + newFileName);
    }
    catch (System.IO.IOException e)
    {
        fileRepeat++;
        goto again;
    }
}
}
}

```

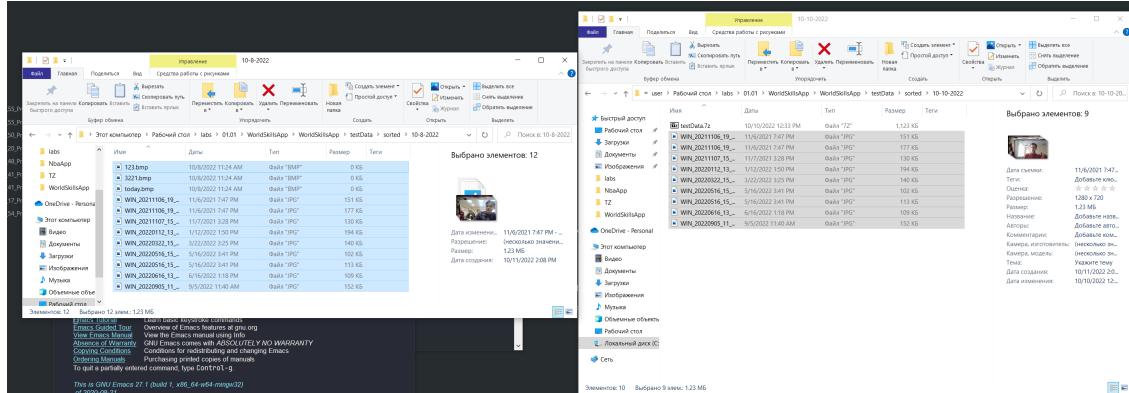
Демонстрация работы:
Содержимое папок:



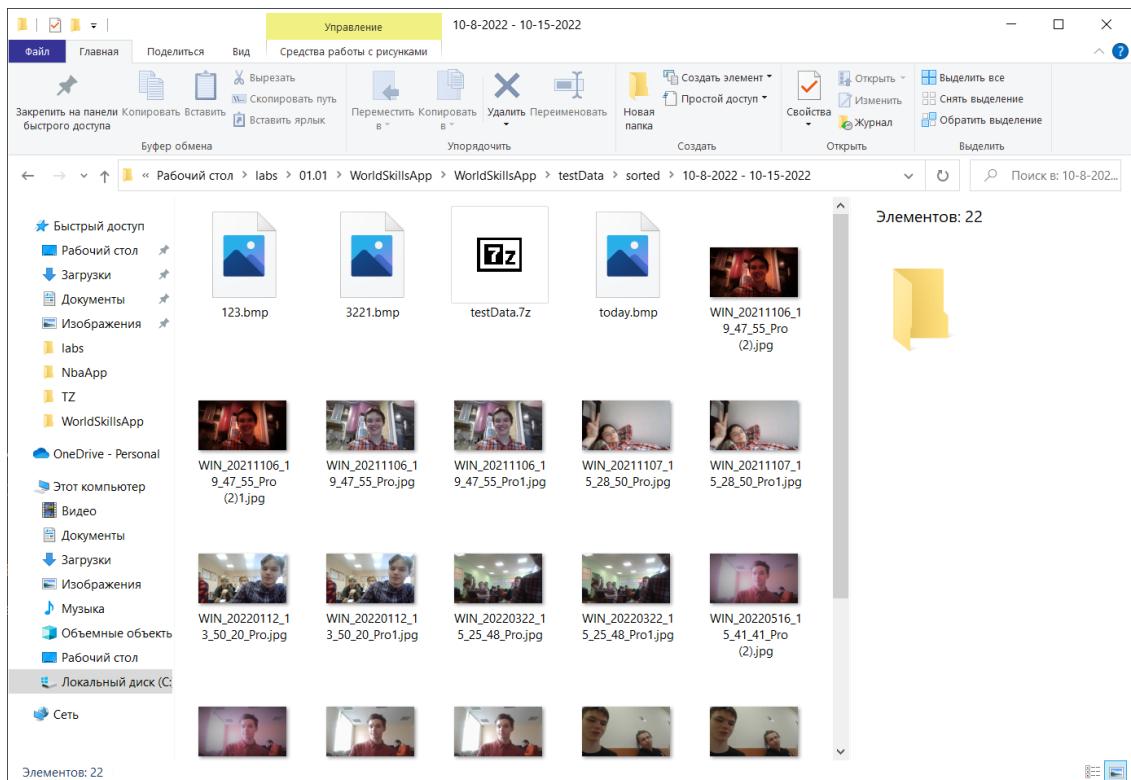
Результат работы:

указав путь до папки з фото(По умолчанию - данная директория)
c:\Users\user\Desktop\labs\01.01\WorldSkillsApp\WorldSkillsApp\test\data\дубль\WIN_20211106_19_47_55_Pro.jpg C:\Users\user\Desktop\labs\01.01\WorldSkillsApp\WorldSkillsApp\test\data\WIN_20211106_19_47_55_Pro.jpg
дубликаты:
c:\Users\user\Desktop\labs\01.01\WorldSkillsApp\WorldSkillsApp\test\data\дубль\WIN_20211106_19_47_55_Pro (2).jpg C:\Users\user\Desktop\labs\01.01\WorldSkillsApp\WorldSkillsApp\test\data\WIN_20211106_19_47_55_Pro (2).jpg
c:\Users\user\Desktop\labs\01.01\WorldSkillsApp\WorldSkillsApp\test\data\дубль\WIN_20211106_19_47_55_Pro (3).jpg C:\Users\user\Desktop\labs\01.01\WorldSkillsApp\WorldSkillsApp\test\data\WIN_20211106_19_47_55_Pro (3).jpg
c:\Users\user\Desktop\labs\01.01\WorldSkillsApp\WorldSkillsApp\test\data\дубль\WIN_20211107_15_28_50_Pro.jpg C:\Users\user\Desktop\labs\01.01\WorldSkillsApp\WorldSkillsApp\test\data\WIN_20211107_15_28_50_Pro.jpg
c:\Users\user\Desktop\labs\01.01\WorldSkillsApp\WorldSkillsApp\test\data\дубль\WIN_20201112_13_50_20_Pro.jpg C:\Users\user\Desktop\labs\01.01\WorldSkillsApp\WorldSkillsApp\test\data\WIN_20201112_13_50_20_Pro.jpg
c:\Users\user\Desktop\labs\01.01\WorldSkillsApp\WorldSkillsApp\test\data\дубль\WIN_2020322_15_25_48_Pro.jpg C:\Users\user\Desktop\labs\01.01\WorldSkillsApp\WorldSkillsApp\test\data\WIN_2020322_15_25_48_Pro.jpg
c:\Users\user\Desktop\labs\01.01\WorldSkillsApp\WorldSkillsApp\test\data\дубль\WIN_2020516_15_41_41_Pro (2).jpg C:\Users\user\Desktop\labs\01.01\WorldSkillsApp\WorldSkillsApp\test\data\WIN_2020516_15_41_41_Pro (2).jpg
c:\Users\user\Desktop\labs\01.01\WorldSkillsApp\WorldSkillsApp\test\data\дубль\WIN_2020516_15_41_41_Pro (3).jpg C:\Users\user\Desktop\labs\01.01\WorldSkillsApp\WorldSkillsApp\test\data\WIN_2020516_15_41_41_Pro (3).jpg
c:\Users\user\Desktop\labs\01.01\WorldSkillsApp\WorldSkillsApp\test\data\дубль\WIN_2020616_14_50_19_Pro.jpg C:\Users\user\Desktop\labs\01.01\WorldSkillsApp\WorldSkillsApp\test\data\WIN_2020616_14_50_19_Pro.jpg
c:\Users\user\Desktop\labs\01.01\WorldSkillsApp\WorldSkillsApp\test\data\дубль\WIN_2020995_11_40_54_Pro.jpg C:\Users\user\Desktop\labs\01.01\WorldSkillsApp\WorldSkillsApp\test\data\WIN_2020995_11_40_54_Pro.jpg
c:\Users\user\Desktop\labs\01.01\WorldSkillsApp\WorldSkillsApp\test\data\дубль\WIN_2021106_19_47_55_Pro.jpg C:\Users\user\Desktop\labs\01.01\WorldSkillsApp\WorldSkillsApp\test\data\WIN_2021106_19_47_55_Pro.jpg
небереди період (d - day w - week, m - month)

По дням:

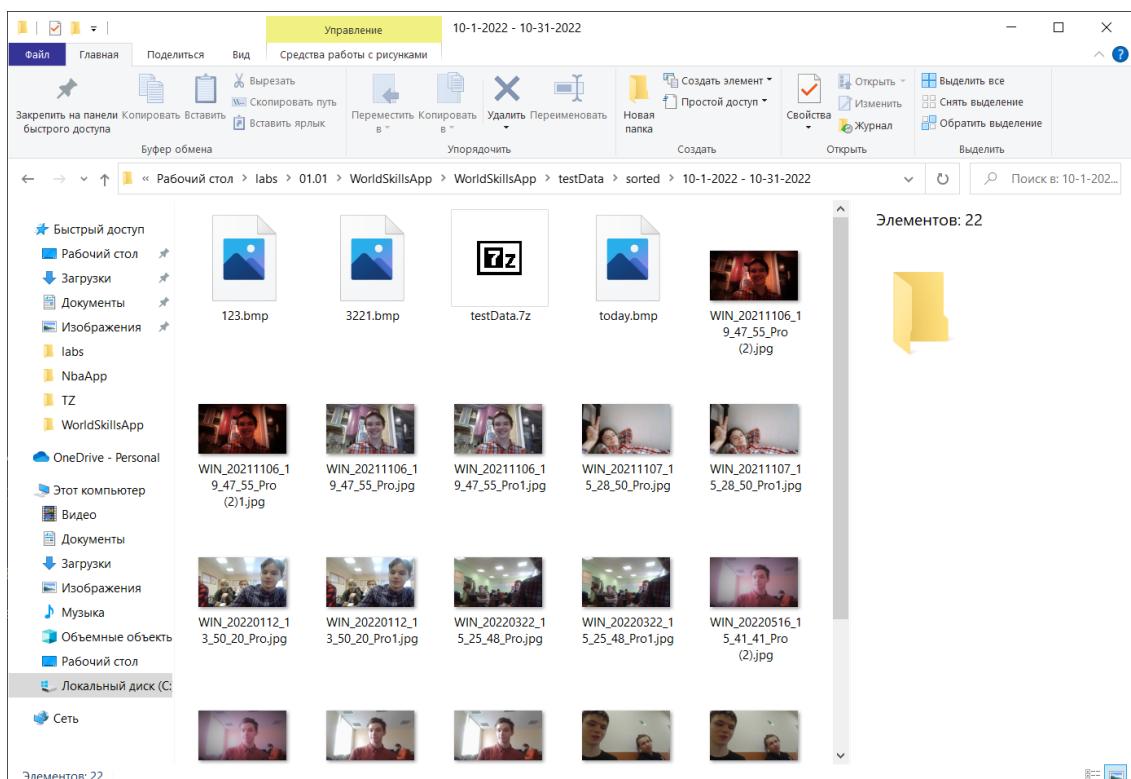


По неделям:



Все картинки, так как они были созданы в эту неделю - главное это название папки(10-8-2022 - 10-15-2022)

По месяцам:



3 Лабораторная работа №3

Тема: Создание собственных классов в C#. Цель работы: получение практических навыков при создании и наследовании классов в C#.

Выполнение работы:

Задание 1. Создание классов по вариантам.

12 вариант

Создать класс квадратная матрица, поля класса – размерность и элементы матрицы. Методы класса: проверка, является ли матрица верхнетреугольной или нижнетреугольной, вывод матрицы. В классе предусмотреть методы: сложение, вычитание, умножение матриц, умножение матрицы на число.

Код:

```
using System;
using System.Collections.Generic;
using System.Linq;

namespace MyClasses;

/* Создать класс квадратная матрица, поля класса - размерность и элементы матрицы.
Методы класса: проверка, является ли матрица верхнетреугольной или нижнетреугольной,
→ вывод матрицы.
В классе предусмотреть методы: сложение, вычитание, умножение матриц, умножение матрицы
→ на число. */

public class SquareMatrix
{
    // Размерность матрицы
    private int _n;

    public int N
    {
        get => _n;
        set
        {
            if (value <= 0)
                throw new Exception("Размерность матрицы должна быть
→ больше 0");
            _n = value;
        }
    }

    // Список содержащий элементы матрицы
    private List<List<int>> _matrixList = null!;

    public List<List<int>> MatrixList
    {
```

```

        get => _matrixList;
        set
        {
            if (value.Count != N || value[0].Count != N)
                throw new Exception("Вводимая матрица неподходит по
→  размерности");
            _matrixList = value;
        }
    }

    public List<int> this[int i]
    {
        get { return MatrixList[i]; }
        set
        {
            if (value.Count != N)
                throw new Exception("Вводимая строка неподходит по
→  размерности");
            MatrixList[i] = value;
        }
    }

    public SquareMatrix(int n)
    {
        N = n;
        var matrixList = new List<List<int>>();
        for (int i = 0; i < n; i++)
        {
            matrixList.Add(new List<int>());
            for (int j = 0; j < n; j++)
            {
                matrixList.Last().Add(0);
            }
        }

        MatrixList = matrixList;
    }

    public SquareMatrix(int n, List<List<int>> matrixList)
    {
        N = n;
        MatrixList = matrixList;
    }

    public override string ToString()
    {
        string output = "[\n";
        foreach (var list in MatrixList)

```

```

{
    output += '\t' + String.Join('\t', list) + '\n';
}

output += "]";
return output;
}

// Это верхняя треугольная матрица?
public bool IsUpperTriangular()
{
    for (int i = 0; i < N; i++)
    {
        for (int j = 0; j < i; j++)
        {
            if (MatrixList[i][j] != 0)
                return false;
        }
    }

    return true;
}

// Это нижняя треугольная матрица?
public bool IsBottomTriangular()
{
    for (int i = 0; i < N; i++)
    {
        for (int j = N - 1; j > i; j--)
        {
            if (MatrixList[i][j] != 0)
                return false;
        }
    }

    return true;
}

// Сложение матриц
public static SquareMatrix operator +(SquareMatrix squareMatrix, SquareMatrix
→ squareMatrix2)
{
    if (squareMatrix2.N != squareMatrix.N)
        throw new Exception("Матрицы не совпадают по размерности");
    var newMatrix = new SquareMatrix(squareMatrix.N);
    for (int i = 0; i < squareMatrix.N; i++)
    {
        for (int j = 0; j < squareMatrix.N; j++)

```

```

                newMatrix[i][j] = squareMatrix[i][j] +
→    squareMatrix2[i][j];
            }

        return newMatrix;
    }

    // Вычитание матриц
    public static SquareMatrix operator -(SquareMatrix squareMatrix, SquareMatrix
→ squareMatrix2)
{
    if (squareMatrix2.N != squareMatrix.N)
        throw new Exception("Матрицы не совпадают по размерности");
    var newMatrix = new SquareMatrix(squareMatrix.N);
    for (int i = 0; i < squareMatrix.N; i++)
    {
        for (int j = 0; j < squareMatrix.N; j++)
            newMatrix[i][j] = squareMatrix[i][j] -
→    squareMatrix2[i][j];
    }

    return newMatrix;
}

// Умножение матрицы на число
public static SquareMatrix operator *(SquareMatrix squareMatrix, int n)
{
    var newMatrix = new SquareMatrix(squareMatrix.N);
    for (int i = 0; i < squareMatrix.N; i++)
    {
        for (int j = 0; j < squareMatrix.N; j++)
            newMatrix[i][j] = squareMatrix[i][j] * n;
    }

    return newMatrix;
}

// Перемножение матриц
public static SquareMatrix operator *(SquareMatrix squareMatrix, SquareMatrix
→ squareMatrix2)
{
    if (squareMatrix2.N != squareMatrix.N)
        throw new Exception("Матрицы не совпадают по размерности");
    var newMatrix = new SquareMatrix(squareMatrix.N);
    for (int i = 0; i < squareMatrix.N; i++)
    {
        for (int j = 0; j < squareMatrix.N; j++)
        {

```

```

        int cellValue = 0;
        for (int j1 = 0; j1 < squareMatrix.N; j1++)
        {
            cellValue += squareMatrix[i][j1] *
→    squareMatrix2[j1][j];
        }

        newMatrix[i][j] = cellValue;
    }

    return newMatrix;
}
}

```

Код тестов:

```

using System;
using System.Collections.Generic;
using NUnit.Framework;

namespace MyClasses;

public class SquareMatrixTests
{
    [Test]
    public void InitTest()
    {
        var matrix = new SquareMatrix(2);
        List<List<int>> myMatrixList = new List<List<int>>();
        myMatrixList.Add(new List<int>() {0, 0});
        myMatrixList.Add(new List<int>() {0, 0});
        Assert.AreEqual(matrix.MatrixList, myMatrixList);
        myMatrixList[0][0] = 1;
        var matrix2 = new SquareMatrix(2, myMatrixList);
        Assert.AreEqual(matrix2.MatrixList, myMatrixList);
    }

    [Test]
    public void StringTest()
    {
        Assert.AreEqual(new SquareMatrix(1).ToString(), "[\n\t0\n]");
        Console.WriteLine(new SquareMatrix(10).ToString());
    }

    [Test]
    public void TriangularTest()
    {
        List<List<int>> myMatrixList = new List<List<int>>();

```

```

        myMatrixList.Add(new List<int>() {1, 0});
        myMatrixList.Add(new List<int>() {1, 1});
        var matrix = new SquareMatrix(2, myMatrixList);
        Assert.IsFalse(matrix.IsUpperTriangular());
        Assert.IsTrue(matrix.IsBottomTriangular());
        myMatrixList = new List<List<int>>();
        myMatrixList.Add(new List<int>() {1, 0, 1});
        myMatrixList.Add(new List<int>() {0, 1, 0});
        myMatrixList.Add(new List<int>() {0, 0, 1});
        matrix = new SquareMatrix(3, myMatrixList);
        Assert.IsTrue(matrix.IsUpperTriangular());
        Assert.IsFalse(matrix.IsBottomTriangular());
    }

    class MathTests
    {
        [Test]
        public void MatrixAdditionTest()
        {
            List<List<int>> myMatrixList = new List<List<int>>();
            myMatrixList.Add(new List<int>() {1, 0});
            myMatrixList.Add(new List<int>() {1, 1});
            var matrix = new SquareMatrix(2, myMatrixList);

            Assert.AreEqual((matrix + matrix)[0], new List<int>() {2, 0});
            Assert.AreEqual((matrix + matrix)[1], new List<int>() {2, 2});
            Assert.AreEqual((matrix - matrix - matrix)[0], new List<int>()
                ↳ {-1, 0});
                Assert.AreEqual((matrix - matrix - matrix)[1], new List<int>()
                ↳ {-1, -1});
        }

        [Test]
        public void ScalarMultiplication()
        {
            List<List<int>> myMatrixList = new List<List<int>>();
            myMatrixList.Add(new List<int>() {1, 2});
            myMatrixList.Add(new List<int>() {3, 4});
            var matrix = new SquareMatrix(2, myMatrixList);
            Assert.AreEqual((matrix * 5)[0], new List<int>() {5, 10});
            Assert.AreEqual((matrix * 5)[1], new List<int>() {15, 20});
        }

        [Test]
        public void Multiplication()
        {
            List<List<int>> myMatrixList = new List<List<int>>();
            myMatrixList.Add(new List<int>() {1, 2});

```

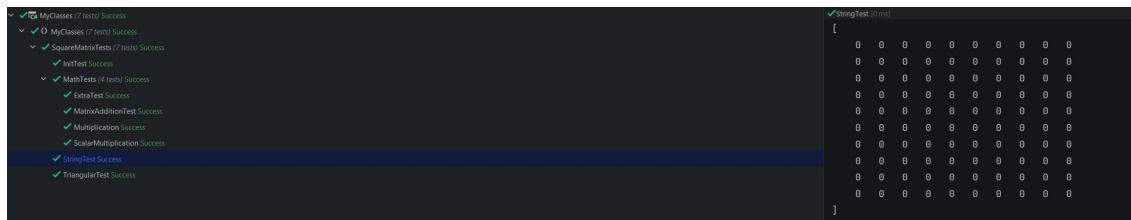
```

        myMatrixList.Add(new List<int>() {3, 4});
        var matrix = new SquareMatrix(2, myMatrixList);
        Assert.AreEqual((matrix * matrix)[0], new List<int>() {7, 10});
        Assert.AreEqual((matrix * matrix)[1], new List<int>() {15, 22});
        myMatrixList = new List<List<int>>();
        myMatrixList.Add(new List<int>() {1, 2, 3});
        myMatrixList.Add(new List<int>() {1, 2, 3});
        myMatrixList.Add(new List<int>() {1, 2, 3});
        matrix = new SquareMatrix(3, myMatrixList);
        Assert.AreEqual((matrix * matrix)[0], new List<int>() {6, 12,
    ↵ 18});
        Assert.AreEqual((matrix * matrix)[1], new List<int>() {6, 12,
    ↵ 18});
        Assert.AreEqual((matrix * matrix)[2], new List<int>() {6, 12,
    ↵ 18});
    }

    [Test]
    public void ExtraTest()
    {
        List<List<int>> myMatrixList = new List<List<int>>();
        myMatrixList.Add(new List<int>() {1, 2});
        myMatrixList.Add(new List<int>() {3, 4});
        var matrix = new SquareMatrix(2, myMatrixList);
        matrix = ((matrix + matrix) * 2 - matrix) * matrix;
        Assert.AreEqual(matrix[0], new List<int>() {21, 30});
        Assert.AreEqual(matrix[1], new List<int>() {45, 66});
    }
}

```

Результаты тестирования:



Задание 2. Создать класс Пароль. Поле класса – пароль. Метод класса – проверка пароля с выводом информационного сообщения: «Пароль верный» или «Пароль неверный». Для простоты пароль будет задаваться программистом в основной программе. Создать класс Надежный пароль, который является потомком класса Пароль и имеет собственный метод анализа надежности пароля:

- Пароль должен состоять не менее чем из 8 символов(слабый)

- Пароль должен содержать как маленькие, так и большие латинские буквы(средний)
- Пароль должен содержать хотя бы одну цифру(хороший)
- Пароль должен содержать хотя бы один символ(!, \$, #, %)(надежный)

В основной программе:

1. подключить модуль с описанными классами;
2. разместить на форме текстовое поле для ввода пароля;
3. в обработчике события кнопки Проверка реализовать работу с созданными классами.

Усложнение. Создать класс Шифр – наследник класса Пароль. Методы класса – Шифрование и Дешифрование пароля с использованием «Шифра Цезаря».

Шифр Цезаря, также известный как шифр сдвига, код Цезаря или сдвиг Цезаря — один из самых простых и наиболее широко известных методов шифрования.

Шифр Цезаря — это вид шифра подстановки, в котором каждый символ в открытом тексте заменяется символом, находящимся на некотором постоянном числе позиций левее или правее него в алфавите. Например, в шифре со сдвигом вправо на 3, А была бы заменена на Г, Б станет Д, и так далее.

В основной программе реализовать следующие функции с использованием класса Шифр:

1. В текстовое поле вводится пароль для шифрования и в зашифрованном виде записывается в текстовый файл.
2. Перед проверкой пароля его необходимо считать из файла и дешифровать.

Код:

- Библиотека PasswordLibrary

- Password.cs (Класс пароль)

```
namespace PasswordLibrary;

public class Password
{
    protected string _password;

    public Password(string password)
    {
```

```

        _password = password;
    }

    public bool CheckPassword(string password)
    {
        return password == _password;
    }
}

```

– PasswordType.cs (перечисление типов паролей)

```

namespace PasswordLibrary;

public enum PasswordType
{
    Bad,
    Weak,
    Medium,
    Good,
    Strong
}

```

– StrongPassword.cs (определение вида пароля)

```

using System.Text.RegularExpressions;

namespace PasswordLibrary;

public class StrongPassword : Password
{
    public Dictionary<PasswordType, string> TypeToPattern = new()
    {
        {PasswordType.Bad, @"[\S]{0,8}"},
        {PasswordType.Weak, @"[\S]{8,}"},
        {PasswordType.Medium, @"(?=.*[A-Z])(?=.*[a-z])(?=.*\S{8,})"},
        {PasswordType.Good,
            @"(?=.*[A-Z])(?=.*[a-z])(?=.*\S{8,})(?=.*[0-9])",
            {PasswordType.Strong,
                @"(?=.*[A-Z])(?=.*[a-z])(?=.*\S{8,})(?=.*[0-9])(?=.*[$#*])"
            };
        }

        public PasswordType CheckPasswordOnStrong()
        {
            var passwordType = PasswordType.Weak;
            foreach (var keyValue in TypeToPattern)
            {
                if (Regex.IsMatch(_password,
                    TypeToPattern[keyValue.Key]))
                    passwordType = keyValue.Key;
            }
            return passwordType;
        }
}

```

```

        public StrongPassword(string password) : base(password)
        {
        }
    }

- EncryptPassword.cs (шифрование пароля)

using System.Text;

namespace PasswordLibrary;

public class EncryptPassword : Password
{
    public EncryptPassword(string password) : base(password)
    {
    }

    public string Encrypt(uint offset)
    {
        offset %= 26;
        var newPass = "";
        foreach (var ch in _password)
        {
            if (ch >= 65 && ch <= 122)
            {
                if (ch <= 90)
                    newPass += char.ToString((char) (ch
→ + offset >= 91
                        ? 65 + offset - (91 - ch)
                        : ch + offset));
                else if (ch >= 97)
                    newPass += char.ToString((char) (ch
→ + offset > 122
                        ? 97 + offset - (123 - ch)
                        : ch + offset));
                else
                    newPass += ch;
            }
            else
            {
                newPass += ch;
            }
        }

        return newPass;
    }

    public string Encrypt(int offset)
    {
        if (offset < 0)
            return Decode((uint) Math.Abs(offset));
        return Encrypt((uint) offset);
    }
}

```

```

    }

    public string Decode(uint offset)
    {
        offset %= 26;

        var newPass = "";
        foreach (var ch in _password)
            if (ch >= 65 && ch <= 122)
            {
                if (ch <= 90)
                    newPass += char.ToString((char) (ch
→ - offset < 65
                     ? 91 - (offset - (ch - 65))
                     : ch - offset));
                else if (ch >= 97)
                    newPass += char.ToString((char) (ch
→ - offset < 97 ? 122 - (offset - (ch - 96)) : ch - offset));
                else
                    newPass += ch;
            }
            else
            {
                newPass += ch;
            }
        }

        return newPass;
    }

    public string Decode(int offset)
    {
        if (offset < 0)
            return Encrypt((uint) Math.Abs(offset));
        return Decode((uint) offset);
    }
}

```

- Тестирование (TestPassword)

- UnitTest1.cs

```

using NUnit.Framework;
using PasswordLibrary;

namespace TestPassword;

public class PasswordsTests
{
    [SetUp]

```

```

public void Setup()
{
}

[TestMethod]
public void StrongPasswordTest()
{
    Assert.AreEqual(new
    StrongPassword("").CheckPasswordOnStrong(), PasswordType.Bad);
    Assert.AreEqual(new
    StrongPassword("12345678").CheckPasswordOnStrong(), PasswordType.Weak);
    Assert.AreEqual(new
    StrongPassword("ASDasdasd").CheckPasswordOnStrong(),
    PasswordType.Medium);
    Assert.AreEqual(new
    StrongPassword("ASDasdasd1").CheckPasswordOnStrong(),
    PasswordType.Good);
    Assert.AreEqual(new
    StrongPassword("ASDasdasd1!").CheckPasswordOnStrong(),
    PasswordType.Strong);
}

[TestMethod]
public void PasswordTest()
{
    Assert.IsTrue(new Password("ok").CheckPassword("ok"));
    Assert.IsFalse(new Password("123").CheckPassword("ok"));
}

[TestMethod]
public void EncryptTest()
{
    Assert.AreEqual(new EncryptPassword("ok").Encrypt(3), "rn");
    Assert.AreEqual(new EncryptPassword("OK").Encrypt(3), "RN");

    Assert.AreEqual(new EncryptPassword("ok1").Encrypt(3),
    "rn1");
    Assert.AreEqual(new EncryptPassword("okB").Encrypt(3),
    "rnB");
    Assert.AreEqual(new EncryptPassword("ok").Encrypt(29),
    "rn");
    Assert.AreEqual(new EncryptPassword("z").Encrypt(1), "a");
    Assert.AreEqual(new EncryptPassword("Z").Encrypt(1), "A");
    Assert.AreEqual(new EncryptPassword("Z").Encrypt(2), "B");
    Assert.AreEqual(new EncryptPassword("a").Decode(1), "z");
    Assert.AreEqual(new EncryptPassword("A").Decode(1), "Z");
}

```

```

        Assert.AreEqual(new EncryptPassword("rn").Decode(29), "ok");
        Assert.AreEqual(new EncryptPassword("A").Decode(0), "A");
        Assert.AreEqual(new EncryptPassword("A").Decode(0), "A");
        Assert.AreEqual(new EncryptPassword("BCDE").Encrypt(-54),
    ↳ "ZABC");
    ↳ // Assert.AreEqual(new EncryptPassword("a").Decode(1),
    ↳ "z"); BCDE
}
}

```

- Приложение с графическим интерфейсом (PasswordC)

- MainWindow.xaml (Разметка)

```

<Window x:Class="PasswordC.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    ↳ action"
        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"
    ↳ mc:Ignorable="d"
        Title="MainWindow" Height="450" Width="800">
<Grid>
    <Grid.RowDefinitions>
        <RowDefinition />
        <RowDefinition />
        <RowDefinition />

    </Grid.RowDefinitions>
    <TextBlock VerticalAlignment="Bottom" Text="{Binding
    ↳ PassStrongMessage}" HorizontalAlignment="Center" />

    <StackPanel Grid.Row="1" Orientation="Horizontal"
    ↳ HorizontalAlignment="Center" VerticalAlignment="Center">
        <TextBlock>Пароль 1:</TextBlock>
        <TextBox Text="{Binding Password1}" Margin="10, 0"
    ↳ Width="100" />
        <TextBlock Margin="10, 0">Пароль 2:</TextBlock>
        <TextBox Text="{Binding Password2}" Margin="10, 0"
    ↳ Width="100" />
        <Button Click="EncryptOnClick">Шифровать</Button>
        <Button Margin="10, 0"
    ↳ Click="CheckOnClick">Проверить</Button>

    </StackPanel>

```

```

        <TextBlock Grid.Row="2" VerticalAlignment="Top"
           → Text="{Binding ErrorMessage}" Foreground="Red"
           → FontSize="20"
                           HorizontalAlignment="Center" />
    </Grid>
</Window>

```

– MainWindow.xaml.cs (Код)

```

using System;
using System.Collections.Generic;
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;

namespace PasswordC;

/// <summary>
/// Interaction logic for MainWindow.xaml
/// </summary>
public partial class MainWindow : Window
{
    public PasswordViewModel passwordViewModel = new();

    public MainWindow()
    {
        InitializeComponent();
        DataContext = passwordViewModel;
    }

    private void CheckOnClick(object sender, RoutedEventArgs e)
    {
        passwordViewModel.Decode();
    }

    private void EncryptOnClick(object sender, RoutedEventArgs e)
    {
        passwordViewModel.Encrypt();
    }
}

```

– PasswordViewModel.cs (Логика приложения (ViewModel))

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.IO;
using System.Runtime.CompilerServices;
using System.Windows;
using PasswordLibrary;

namespace PasswordC;

public class PasswordViewModel : INotifyPropertyChanged
{
    public event PropertyChangedEventHandler? PropertyChanged;

    protected void NotifyPropertyChanged(string propertyName)
    {
        if (PropertyChanged != null)
            PropertyChanged(this, new
→ PropertyChangedEventArgs(propertyName));
    }

    public Dictionary<PasswordType, string> typeToMessage = new()
    {
        {PasswordType.Bad, "Ваш пароль ужасен"},
        {PasswordType.Weak, "Ваш пароль плохой"},
        {PasswordType.Medium, "Ваш пароль средней сложности"},
        {PasswordType.Good, "Ваш пароль хороший"},
        {PasswordType.Strong, "Ваш пароль сильный"}
    };

    private string _password1 = "";
    private string _password2 = "";

    public StrongPassword StrongPassword = new("");

    public string Password1
    {
        get => _password1;
        set
        {
            if (value.Trim() != "")
            {
                _password1 = value;
                StrongPassword = new
→ StrongPassword(_password1);
                if
→ (!StrongPassword.CheckPassword(_password2))
            }
        }
    }
}
```

```

    {
        ErrorMessage = "Пароли не
        ↵ совпадают";
        PassStrongMessage = "";
    }
    else
    {
        ErrorMessage = "";
        PassStrongMessage =
        ↵ typeToMessage[StrongPassword.CheckPasswordOnStrong()];
    }
}
else
{
    PassStrongMessage = "";
    ErrorMessage = "";
    _password1 = "";
}

NotifyPropertyChanged("PassStrongMessage");
NotifyPropertyChanged("ErrorMessage");
}

}

public string Password2
{
    get => _password2;
    set
    {
        if (value.Trim() != "")
        {
            _password2 = value;
            if
                (!StrongPassword.CheckPassword(_password2))
            {
                ErrorMessage = "Пароли не
                ↵ совпадают";
                PassStrongMessage = "";
            }
            else
            {
                ErrorMessage = "";
                PassStrongMessage =
                ↵ typeToMessage[StrongPassword.CheckPasswordOnStrong()];
            }
        }
    }
}

```

```

        PassStrongMessage = "";
        ErrorMessage = "";
        _password2 = "";
    }

    NotifyPropertyChanged("PassStrongMessage");
    NotifyPropertyChanged("ErrorMessage");
}

}

public string ErrorMessage { get; set; } = "";
public string PassStrongMessage { get; set; } = "";

public void Encrypt()
{
    if (_password1 == _password2 && _password1.Trim() != "")
        using (StreamWriter writer = new("pass.txt"))
    {
        var offset = new Random().Next(-100, 100);
        // var offset = 3;
        writer.WriteLine(offset);
        writer.Write(new
    ← EncryptPassword(_password1).Encrypt(offset));
    }
}

public void Decode()
{
    if (_password1 == _password2 && _password1.Trim() != "")
        using (StreamReader reader = new("pass.txt"))
    {
        var offset = int.Parse(reader.ReadLine()!);
        var savedPass = new
    ← EncryptPassword(reader.ReadLine()!).Decode(offset);
        ErrorMessage = "";
        if (savedPass != _password1)
            ErrorMessage = "Пароль не совпадает
    ← с сохранённым";
        else
            MessageBox.Show("Поздравляю, вы
    ← угадали пароль:)");
        NotifyPropertyChanged("ErrorMessage");
    }
}
}

```

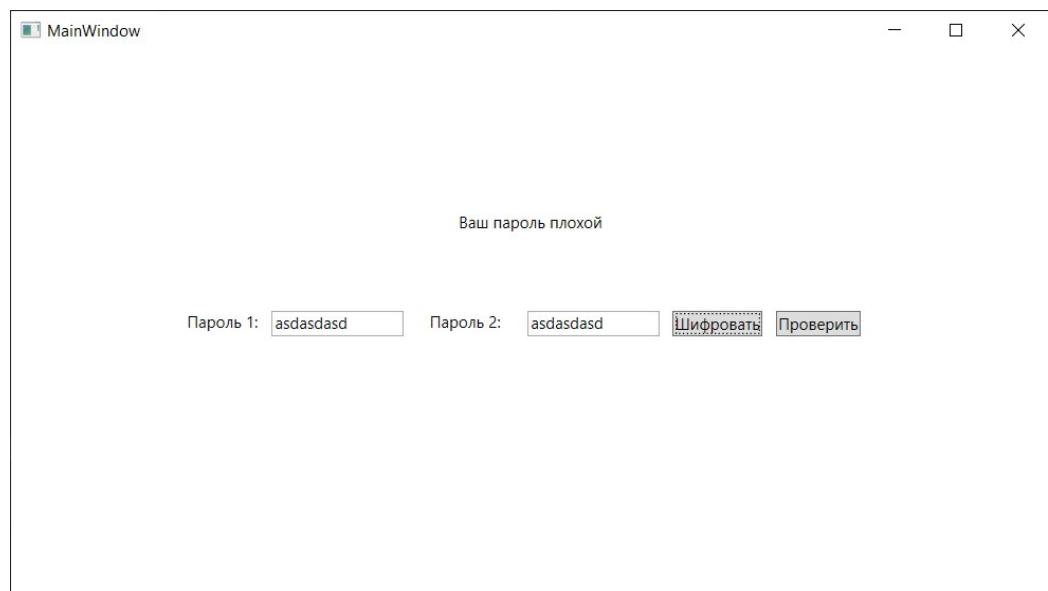
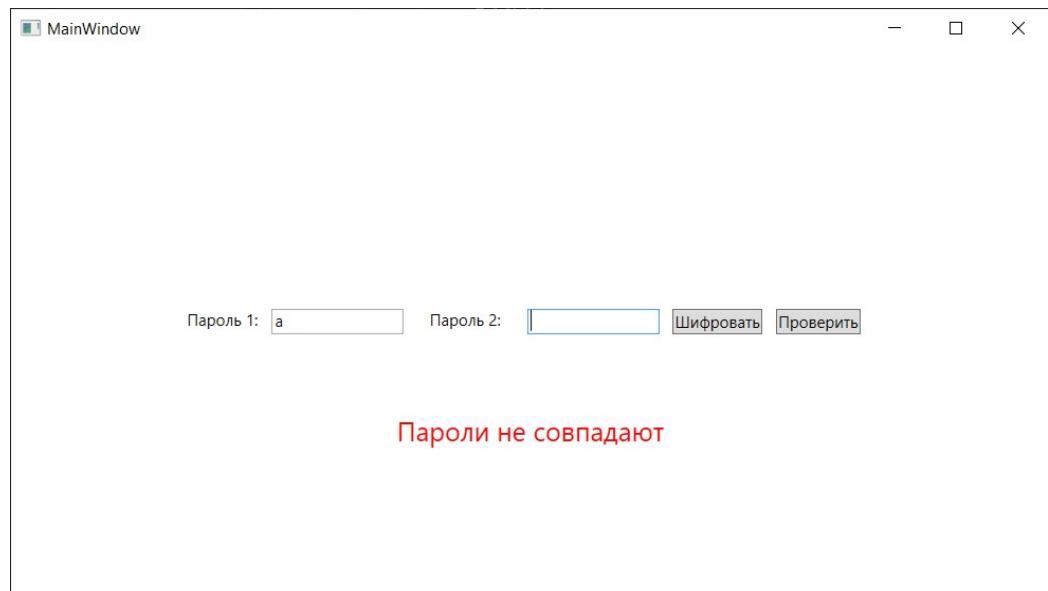
Результаты работы приложения:

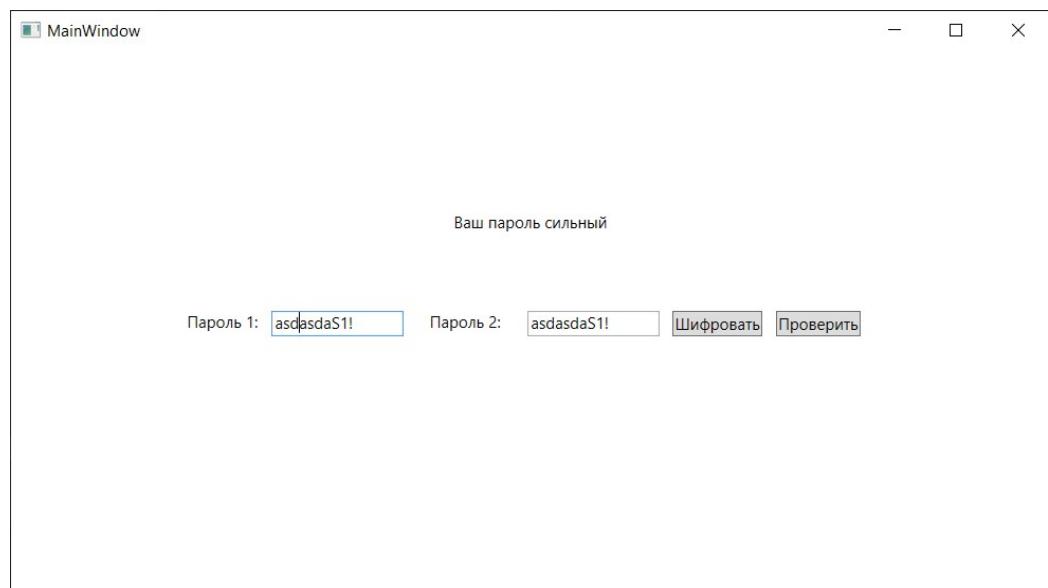
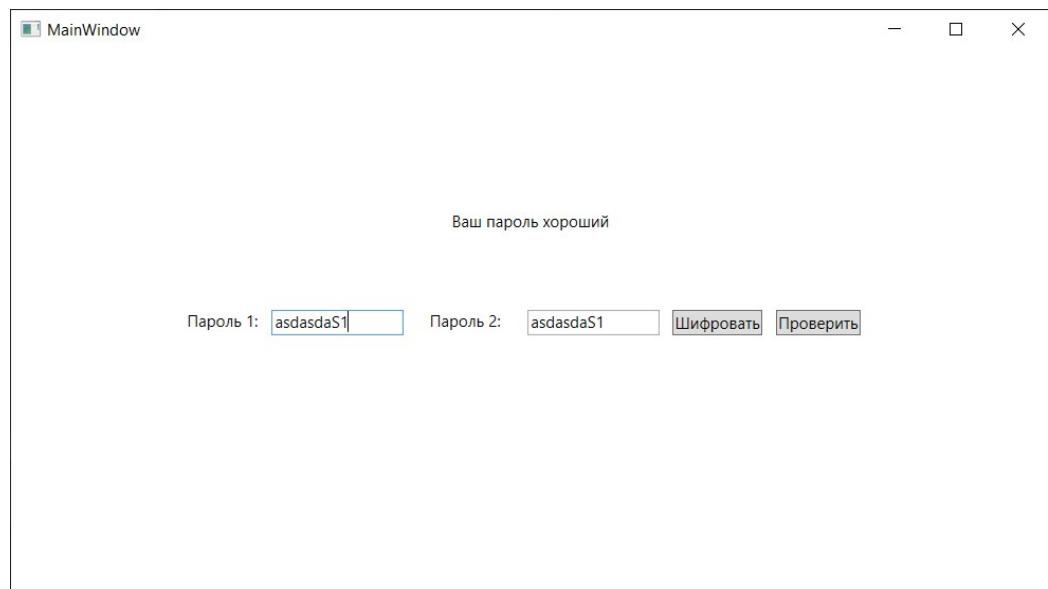
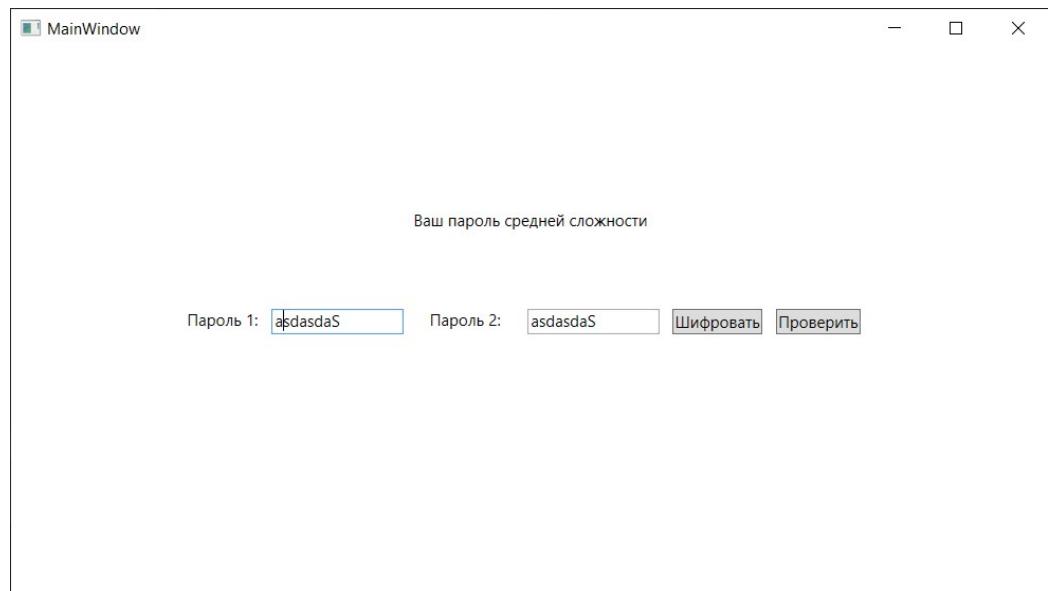
- Тестирование:

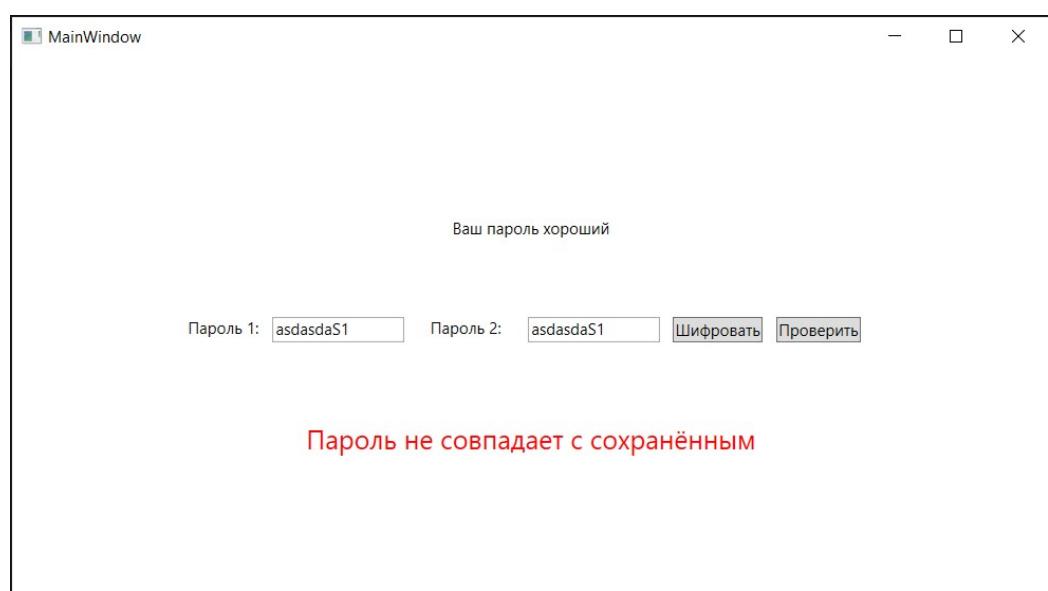
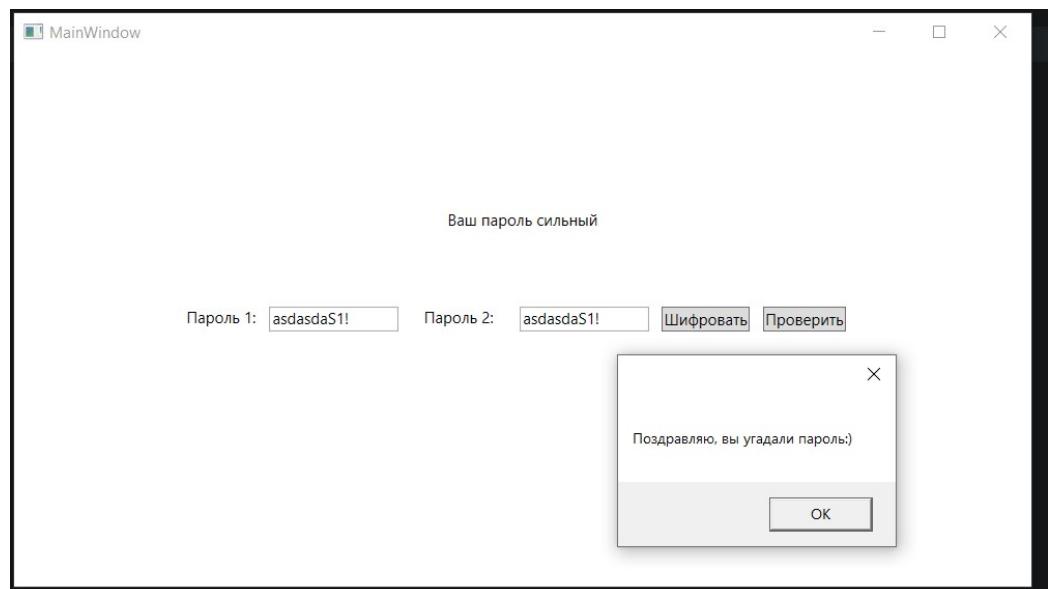
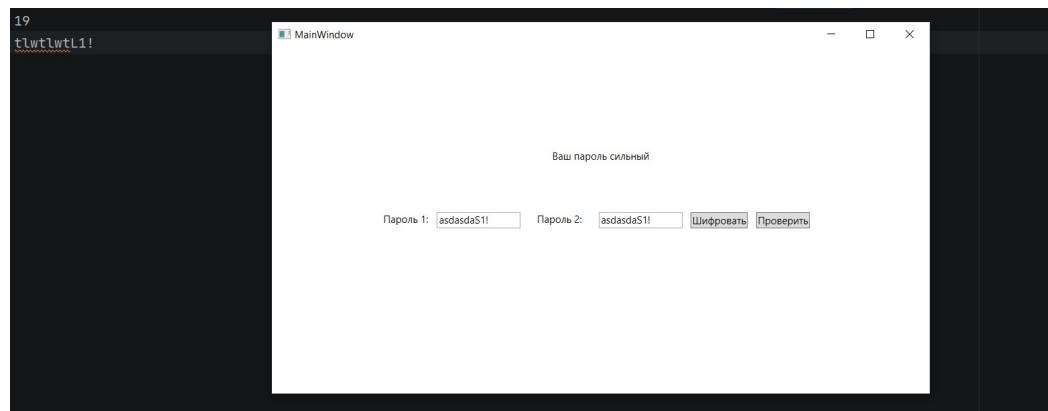
```
✓ c# TestPassword (3 tests) Success
  ✓ () TestPassword (3 tests) Success
    ✓ PasswordsTests (3 tests) Success
      ✓ EncryptTest Success
      ✓ PasswordTest Success
      ✓ StrongPasswordTest Success
```

- Приложение с граф. интерфейсом









Задание 3
Реализовать шифрование и дешифрование строки с помощью шифра Виженера.
Код:
EncryptPassword.cs (Основной класс приложения)

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Reflection.Metadata.Ecma335;

namespace VigenereCipher;

public class EncryptPassword
{
    public List<char> alphabet = "АБВГДЕЁЖЗИЙКЛМНОРСТУФХЦЧШӮЫ҆Я".ToList();
    private string _password;

    public EncryptPassword(string password)
    {
        _password = password;
    }

    private char EncryptCh(char ch, int offset)
    {
        var ind = alphabet.IndexOf(Char.ToUpper(ch));
        if (ind + offset > alphabet.Count - 1)
            return Char.ToUpper(ch)
                ? alphabet[offset - (alphabet.Count - ind)]
                : Char.ToLower(alphabet[offset - (alphabet.Count -
→ ind)]);
        return Char.ToUpper(ch) ? alphabet[ind + offset] :
→ Char.ToLower(alphabet[ind + offset]);
    }

    private char DecodeCh(char ch, int offset)
    {
        var ind = alphabet.IndexOf(Char.ToUpper(ch));
        if (ind - offset < 0)
            return Char.ToUpper(ch)
                ? alphabet[^offset - ind]
                : Char.ToLower(alphabet[^offset - ind]);

        return Char.ToUpper(ch) ? alphabet[ind - offset] :
→ Char.ToLower(alphabet[ind - offset]);
    }

    public string Encrypt(string key)
    {
        string newString = "";
        key = key.ToUpper();
        int keyIndex = 0;
        for (int i = 0; i < _password.Length; i++)
        {

```

```

        if (!alphabet.Contains(char.ToUpper(_password[i])))
        {
            newString += char.ToString(_password[i]);
            continue;
        }

        while (alphabet.IndexOf(key[keyIndex % key.Length]) == -1)
    ↵  keyIndex++;
            newString += char.ToString(EncryptCh(_password[i],
    ↵  alphabet.IndexOf(key[keyIndex++ % key.Length])));
        }

        return newString;
    }

    public string Decode(string key)
    {
        string newString = "";
        key = key.ToUpper();
        int keyIndex = 0;
        for (int i = 0; i < _password.Length; i++)
        {
            if (!alphabet.Contains(char.ToUpper(_password[i])))
            {
                newString += char.ToString(_password[i]);
                continue;
            }

            while (alphabet.IndexOf(key[keyIndex % key.Length]) == -1)
    ↵  keyIndex++;
                newString += char.ToString(DecodeCh(_password[i],
    ↵  alphabet.IndexOf(key[keyIndex++ % key.Length])));
            }

            return newString;
        }
    }
}

```

EncryptClassTests.cs (Тесты над классом выше)

```

using NUnit.Framework;

namespace VigenereCipher;

[TestFixture, Description("Тестирования класса EncryptPassword, который шифрует русский
    текст шифром Веженера"),
Author("Pankov Vasya")]
public class EncryptClassTests
{
    [SetUp]

```

```

public void Setup()
{
}

[TestFixture, Description("Проверка шифрования")]
public class EncryptTests
{
    [Test, Description("Проверка шифрования, при наличии там чисел")]
    public void EncryptWithNums()
    {
        Assert.AreEqual(new EncryptPassword("123").Encrypt("A"), "123");
        Assert.AreEqual(new EncryptPassword("Вася123").Encrypt("Б"), ,
← "Г6тa123");
    }

    [Test, Description("Обычная проверка шифрования")]
    public void EncryptBase()
    {
        Assert.AreEqual(new EncryptPassword("Вася").Encrypt("A"),
← "Вася");
        Assert.AreEqual(new EncryptPassword("Вася").Encrypt("Б"),
← "Гбта");
        Assert.AreEqual(new EncryptPassword("Вася").Encrypt("БА"),
← "Гата");
        Assert.AreEqual(new EncryptPassword("Вася").Encrypt("AAAA"),
← "Вася");
        Assert.AreEqual(new EncryptPassword("Вася").Encrypt("АБВГ"),
← "Вбув");
    }

    [Test, Description("Шифрование с латиницей")]
    public void EncryptWithLatin()
    {
        Assert.AreEqual(new EncryptPassword("ВасяAbc").Encrypt("A"),
← "ВасяAbc");
        Assert.AreEqual(new EncryptPassword("ВасяAbc").Encrypt("Б"),
← "ГбтаAbc");
    }

    [Test]
    public void EncryptExtra()
    {
        Assert.AreEqual(new EncryptPassword("ПРОГРАММИСТЫ УЕХАЛИ В
← КОВОРКИНГ").Encrypt("АГДЕВСЕАГДЕВСЕ АГДЕВСЕ АГДЕВСЕ АГД"), "ПУТЗТССМЛХЧЭ ЕЙХГПН Д
← БУВСФПКЯЗ");
        Assert.AreEqual(new EncryptPassword("ИГРА В ДОМИНО
← УВЛЕКАТЕЛЬНА").Encrypt("ТЕСТ"), "ЫЗВТ Ф ИАЯЛТА ЁФРЦЭТЧЦЮОТС");
    }
}

```

```

    }

    [TestFixture, Description("Проверка декодирования")]
    public class DecodeTests
    {
        [Test, Description("Обычная проверка")]
        public void DecodeBase()
        {
            Assert.AreEqual(new EncryptPassword("Вася").Decode("A"), "Вася");
            Assert.AreEqual(new EncryptPassword("Гбта").Decode("B"), "Вася");
            Assert.AreEqual(new EncryptPassword("Гатя").Decode("BA"),
                           "Вася");
            Assert.AreEqual(new EncryptPassword("Вася").Decode("AAAA"),
                           "Вася");
            Assert.AreEqual(new EncryptPassword("B6yB").Decode("ABBG"),
                           "Вася");
        }

        [Test, Description("Декодирование с числами")]
        public void DecodeWithNums()
        {
            Assert.AreEqual(new EncryptPassword("123").Decode("A"), "123");
            Assert.AreEqual(new EncryptPassword("Гбта123").Decode("B"),
                           "Вася123");
        }

        [Test, Description("Декодирование с латиницей")]
        public void EncryptWithLatin()
        {
            Assert.AreEqual(new EncryptPassword("ГбтаAbc").Decode("B"),
                           "ВасяAbc");
            Assert.AreEqual(new EncryptPassword("ДвябAbc").Decode("B"),
                           "ВасяAbc");
        }

        [Test]
        public void EncryptExtra()
        {
            Assert.AreEqual(new EncryptPassword("ПУТЗТССМЛХЧЭ ЕЙХГПН Д
                                         ЪВСФПКЯЗ").Decode("АГДЕВСЕАГДЕВСЕ АГДЕВСЕ АГДЕВСЕ АГД"),
                           "ПРОГРАММИСТЫ УЕХАЛИ В
                           КОВОРКИНГ");
            Assert.AreEqual(new EncryptPassword("ЫЗВТ Ф ИАЯыта
                                         ёФРЦЭТЧЮОТС").Decode("ТЕСТ"),
                           "ИГРА В ДОМИНО УВЛЕКАТЕЛЬНА");
        }
    }
}

```

Тесты:

The screenshot shows a test results window with the following structure:

- ✓ C# VigenereCipher (8 tests) Success
 - ✓ {} VigenereCipher (8 tests) Success
 - ✓ EncryptClassTests (8 tests) Success: Test not run
 - ✓ DecodeTests (4 tests) Success
 - ✓ DecodeBase Success
 - ✓ DecodeWithNums Success
 - ✓ EncryptExtra Success
 - ✓ EncryptWithLatin Success
 - ✓ EncryptTests (4 tests) Success
 - ✓ EncryptBase Success
 - ✓ EncryptExtra Success
 - ✓ EncryptWithLatin Success
 - ✓ EncryptWithNums Success

4 Лабораторная работа № 4

Тема: Создание консольной заставки(игры) с использованием абстрактного класса(классов).

Цель работы: получение практических навыков при создании классов на базе абстрактного класса.

4.1 Часть 1

Задание.

Создать абстрактный класс Monster со следующими полями: x,y – координаты точки на экране; weight – вес, с которым рождается Monster(использовать спецификатор private). Для работы с полями объекта использовать свойства X,Y,Weight . Данный класс должен содержать метод BeBorn с параметрами. В качестве параметров выступают координаты точки и вес. Значение поля weight устанавливается случайным образом.

Создать потомок созданного класса с именем Dog с собственными полями color(цвет собаки),face(отображение). Доступ к полям реализовать через свойства. В конструкторе данного класса реализовать появление собаки либо желтого, либо темно-желтого цвета.

Создать еще одного потомка созданного класса с именем Hedgehog с собственными полями mcolor(цвет), mface(отображение). Доступ к полям реализовать через свойства.

А) В основной программе с использованием цикла создать какое-то количество экземпляров объекта Dog и Hedgehog по очереди. Координаты x и y задаются случайным образом. При создании экземпляров объекта Dog учесть следующее: если вес при рождении больше 55, то собака закрашивается в фиолетовый цвет. Для экземпляров объекта Hedgehog всегда устанавливать цвет серый.

Б) Создать список объектов класса Dog, используя класс List. Для каждого объекта задать траекторию движения. Организовать цикл с движением объектов. Примерный вид заставки с динамическими объектами можно посмотреть в файле Заставка.exe.

Код:

1. Класс Monster

```
namespace MonsterDog;

public abstract class Monster
{
    protected int _weight;
    public int X, Y;

    public abstract void BeBorn(int x, int y, int weight);
    public abstract Task Move(int dx, int dy);
}
```

2. Класс Dog

```
namespace MonsterDog;

public class Dog : Monster
{
    public ConsoleColor Color;
    public const char Face = '@';

    public Dog()
    {
        byte r = (byte) new Random().Next(10);
        Color = r > 5 ? ConsoleColor.Yellow : ConsoleColor.Green;
    }
}
```

```

public override void BeBorn(int x, int y, int weight)
{
    (X, Y, _weight) = (x, y, weight);
    if (weight > 55)
        Color = ConsoleColor.Gray;

    Utils.DrawSymbol(Face, x, y, Color);
}

public override Task Move(int dx = 0, int dy = 1)
{
    (X, Y) = (X + dx, Y + dy);
    if (X > Console.WindowWidth - 1)
        X = 0 + dx;
    if (Y > Console.WindowHeight)
        Y = 0 + dy;
    if (X < 0)
        X = Console.BufferWidth + dx;
    if (Y < 0)
        Y = Console.WindowHeight + dy;
    Utils.DrawSymbol(Face, X, Y, Color);
    return Task.CompletedTask;
}
}

```

3. Класс Hedgehog

```

using System.Drawing;

namespace MonsterDog;

public class Hedgehog : Monster
{
    public ConsoleColor mColor = ConsoleColor.Gray;
    public char mFace = '*';

    public override void BeBorn(int x, int y, int weight)
    {
        (X, Y, _weight) = (x, y, weight);
        Utils.DrawSymbol(mFace, x, y, mColor);
    }

    public override Task Move(int dx = 1, int dy = 0)
    {
        (X, Y) = (X + dx, Y + dy);
        if (X > Console.WindowWidth - 1)
            X = 0 + dx;
        if (Y > Console.WindowHeight)
            Y = 0 + dy;
    }
}

```

```

        if (X < 0)
            X = Console.BufferWidth + dx;
        if (Y < 0)
            Y = Console.BufferHeight + dy;
        Utils.DrawSymbol(mFace, X, Y, mColor);
        return Task.CompletedTask;
    }
}

```

4. Вспомогательный класс Utils

```

using System.Drawing;

namespace MonsterDog;

public static class Utils
{
    public static void DrawSymbol(char sym, int x, int y, ConsoleColor color)
    {
        Console.SetCursorPosition(x, y);
        var defaultColor = Console.BackgroundColor;
        Console.ForegroundColor = color;
        Console.Write(sym);
        Console.ForegroundColor = defaultColor;
        Console.SetCursorPosition(0, 0);
    }
}

```

5. Основная программа

```

using System.Reflection.Metadata.Ecma335;
using MonsterDog;

Random random = new Random();
Console.Clear();

(int, int) WindowsSize = (Console.WindowWidth, Console.WindowHeight);

var task = Task.Run(async delegate
{
    HashSet<Dog> dogs = new();
    HashSet<Hedgehog> hdogs = new();
    restart:
    Console.Clear();

```

```

        hdogs.Clear();
        dogs.Clear();
        for (int i = 0; i < (WindowSize.Item1 * WindowsSize.Item2) / 100; i++)
        {
            dogs.Add(new Dog());
            regen1:
            (int rx, int ry) = (random.Next(0, Console.WindowWidth),
→ random.Next(0, Console.WindowHeight));
            if (dogs.Count(dog => dog.X == rx && dog.Y == ry) == 1) goto
→ regen1;
            dogs.Last().BeBorn(random.Next(0, Console.WindowWidth),
→ random.Next(0, Console.WindowHeight), random.Next(10, 100));
        }

        for (int i = 0; i < (WindowSize.Item1 * WindowsSize.Item2) / 150; i++)
        {
            hdogs.Add(new Hedgehog());
            regen2:
            (int rx, int ry) = (random.Next(0, Console.WindowWidth),
→ random.Next(0, Console.WindowHeight));
            if (dogs.Count(dog => dog.X == rx && dog.Y == ry) == 1 ||
→ hdogs.Count(dog => dog.X == rx && dog.Y == ry) == 1) goto regen2;
            hdogs.Last().BeBorn(random.Next(0, Console.WindowWidth),
→ random.Next(0, Console.WindowHeight), random.Next(10, 100));
        }

        while (true)
        {
            if (WindowSize.Item1 != Console.WindowWidth || WindowsSize.Item2
→ != Console.WindowHeight)
            {
                WindowsSize = (Console.WindowWidth, Console.WindowHeight);
                goto restart;
            }
            Console.Clear();
            var task = Task.Run(async delegate
            {
                foreach (var dog in dogs)
                {
                    await dog.Move(random.Next(-5, 5),
→ random.Next(-5, 5));
                }
            });
            var task2 = Task.Run(async delegate
            {
                foreach (var hdog in hdogs)
                {
                    await hdog.Move(random.Next(-1, 3),
→ random.Next(-1, 3));
                }
            });
        }
    }
}

```

```

        }
    });

    await Task.WhenAll(task, task2);
    await Task.Delay(1);
}

};

task.Wait();

```

Демонстрация работы программы:



4.2 Часть 2

Задание.

Необходимо реализовать собственную заставку или игру, с помощью абстрактных классов и возможностей работы.

Я реализовал свою версию игры "Три в ряд".

Реализованный функционал:

1. Меню для игры
2. Автоматическое масштабирование интерфейса от размера консоли
3. Можно создавать поле любого размера и любым количеством игровых элементов(которые ставятся в три в ряд)
4. Игра работает довольно быстро
5. Игра бесконечная, так как пока нет цели и ограничений, поэтому в неё можно играть, чтобы просто бороться со стрессом.

Код игры:

1. Utils - вспомогательный абстрактный класс, который содержит в себе утилиты

```
namespace SuperGame;

public static class Utils
{
    public static ConsoleColor defaultBgColor;
    public static ConsoleColor defaultFgColor;

    public static void WriteLinesOnPos(int x, int y, string text,
        ConsoleColor fgColor = ConsoleColor.White, ConsoleColor bgColor =
        ConsoleColor.Black)
    {
        Console.BackgroundColor = bgColor;
        Console.ForegroundColor = fgColor;
        foreach (var row in text.Split('\n'))
        {
            Console.SetCursorPosition(x, y++);
            Console.WriteLine(row);
        }
        Console.BackgroundColor = defaultBgColor;
        Console.ForegroundColor = defaultFgColor;
    }
}
```

2. IScreen - интерфейс, который сделан для работ экранов

```
namespace SuperGame;

public interface IScreen
{
    void CheckInput();
    void Display();
    void Finish();
    void OnReConfigure();

    void Start();
}
```

3. GameEngineStatus - перечисление всех возможных вариаций состояний класса GameEngine

```
namespace SuperGame;

public enum GameEngineStatus
{
    End,
    NotStarted,
    Playing,
    Quit,
    LeaderBoard
}
```

4. GameEngine - класс, который руководит всей игрой и перемещениями между экранами

```
using System.Diagnostics;

namespace SuperGame;

public class GameEngine
{
    private GameEngineStatus _gameEngineStatus = GameEngineStatus.NotStarted;
    private Menu _menu;
    private Game _game;

    private Task _consoleSizeTask;

    public GameEngineStatus GameEngineStatus
    {
        get => _gameEngineStatus;
        set
        {
```

```

        _gameEngineStatus = value;
        switch (value)
        {
            case GameEngineStatus.Quit:
                // Сохраняем что-то
                _menu.Finish();
                break;
            case GameEngineStatus.Playing:
                _menu.Finish();
                // consoleSizeTask = Task.Run(() =>
    ↵    CheckConsoleSize());
                _game = new Game(this);
                _game.Start();
                break;
            case GameEngineStatus.NotStarted:
                _menu = new Menu(this);
                _menu.Start();
                break;
        }
    }

private byte _moves = Constants.MAX_MOVES;

public byte Moves
{
    get => _moves;
    set
    {
        if (value <= 0)
        {
            _moves = value;
            GameEngineStatus = GameEngineStatus.End;
        }
        else if (value > Constants.MAX_MOVES)
            _moves = Constants.MAX_MOVES;

        _moves = value;
    }
}

public GameEngine()
{
    Utils.defaultBgColor = ConsoleColor.Black;
    Utils.defaultFgColor = ConsoleColor.White;
    Console.BackgroundColor = ConsoleColor.Black;
    Console.ForegroundColor = ConsoleColor.White;
    _consoleSizeTask = Task.Run(() => CheckConsoleSize());
}

```

```

        _menu = new Menu(this);
        _menu.Start();
    }

    public void Move()
    {
    }

    private void CheckConsoleSize()
    {
        (int cWidth, int cHeight) = (Console.WindowWidth,
→ Console.WindowHeight);
        while (true)
        {
            if (cWidth != Console.WindowWidth || cHeight !=
→ Console.WindowHeight)
            {
                switch (_gameEngineStatus)
                {
                    case GameEngineStatus.NotStarted:
                        _menu.OnReConfigure();
                        break;
                    case GameEngineStatus.Playing:
                        _game.Display();
                        break;
                }
                (cWidth, cHeight) = (Console.WindowWidth,
→ Console.WindowHeight);
            }
            Thread.Sleep(50);
        }
    }
}

```

5. MenuItem - record или data class, который позволяет удобно организовать Menu

```

namespace SuperGame;

public record MenuItem(string Name, GameEngineStatus Action);

```

6. Menu - класс, который отвечает за работу экрана меню.

```

using System.Threading;
using System.Threading.Tasks;

```

```

namespace SuperGame;

public class Menu: IScreen
{
    // new MenuItem("Таблица Лидеров", GameEngineStatus.LeaderBoard)
    public List<MenuItem> MenuItems = new ()
        {new MenuItem("Начать играть", GameEngineStatus.Playing), new
    ↵ MenuItem("Выйти", GameEngineStatus.Quit)};

    public string Logo = "";
    private GameEngine _gameEngine;
    MenuItem _chosenItem;
    int _cWidth, _cHeight;
    (int X, int Y) _pos = (0, 0);
    private Task _inputTask;
    private bool _isFinish;

    public Menu(GameEngine gameEngine)
    {
        (_cWidth, _cHeight) = (Console.WindowWidth, Console.WindowHeight);
        _chosenItem = MenuItems.First();
        try
        {
            Logo = File.ReadAllText("res\\Hello.txt");
        }
        catch (FileNotFoundException e)
        {
            Logo = "Поиграем без логотипа";
        }

        _gameEngine = gameEngine;
    }

    public void Start()
    {
        Display();
        _inputTask = Task.Run(() => CheckInput());
        _inputTask.Wait();
    }

    public void Finish()
    {
        _isFinish = true;
    }

    public void OnReConfigure()

```

```

{
    Display();
}
public void CheckInput()
{
    while (_gameEngine.GameEngineStatus ==
        GameEngineStatus.NotStarted)
    {

        while (!Console.KeyAvailable)
        {
            if (_isFinish) return;
        }

        var key = Console.ReadKey();
        switch (key.Key)
        {
            case ConsoleKey.DownArrow:
            case ConsoleKey.S:
                if (_chosenItem == MenuItems.Last())
                    _chosenItem = MenuItems.First();
                else
                {
                    _chosenItem =
                    MenuItems[MenuItems.IndexOf(_chosenItem) + 1];
                }

                DisplayMenu();
                break;
            case ConsoleKey.UpArrow:
            case ConsoleKey.W:
                if (_chosenItem == MenuItems.First())
                    _chosenItem = MenuItems.Last();
                else
                {
                    _chosenItem =
                    MenuItems[MenuItems.IndexOf(_chosenItem) - 1];
                }

                DisplayMenu();
                break;
            case ConsoleKey.Enter:
                _gameEngine.GameEngineStatus =
                _chosenItem.Action;
                return;
            break;
        }
    }
}

```

```

        }

    }

    public void Display()
    {
        (_cWidth, _cHeight) = (Console.WindowWidth, Console.WindowHeight);
        // Отображение логотипа
        Console.Clear();
        _pos = (_cWidth / 2 - (Logo.Split('\n')).MaxBy(s => s.Length) ??
→ "") .Length / 2, _cHeight / 10);

        Utils.WriteLineOnPos(_pos.X, _pos.Y, Logo);

        DisplayMenu();
    }

    void DisplayMenu()
    {
        // Отображение вариантов выбора в меню
        var defaultBgColor = Console.BackgroundColor;
        var defaultFgColor = Console.ForegroundColor;

        _pos = (0, _cHeight / 2);
        foreach (var item in MenuItems)
        {
            Console.SetCursorPosition(_cWidth / 2 - item.Name.Length
→ / 2, _pos.Y++);
            if (item == _chosenItem)
            {
                Console.BackgroundColor = ConsoleColor.White;
                Console.ForegroundColor = ConsoleColor.Black;
                Console.WriteLine(item.Name);
                (Console.BackgroundColor,
→ Console.ForegroundColor) = (defaultBgColor, defaultFgColor);
            }
            else
            {
                Console.WriteLine(item.Name);
            }
        }

        Console.SetCursorPosition(_cWidth - 1, _cHeight - 2);
    }
}

```

7. GameItem - абстрактный класс отвечающий за работу всех элементов в игре

```

namespace SuperGame;

public abstract class GameItem : ICloneable
{
    public (int x, int y) Position = (0, 0);
    public char DisplayChar;
    public ConsoleColor Color;
    public int[,] Board;

    public abstract string Display(int sideSize);

    public abstract void Move(int dx, int dy, int i, int dy1, Game game);

    public GameItem(char displayChar, ConsoleColor color)
    {
        (DisplayChar, Color) = (displayChar, color);
    }

    public object Clone()
    {
        return MemberwiseClone();
    }
}

```

8. BasicItem - класс наследующий абстрактный GameItem и реализует работу стандартного элемента в игре

```

namespace SuperGame;

public class BasicItem : GameItem
{

    public override string Display(int SideSize)
    {
        string result = "";
        for (int i = 0; i < SideSize; i++)
        {
            result += new string(DisplayChar, SideSize);
            // Убираем последний перенос строки, чтобы не ломать
            ← систему, лишним переносом
            if (i != SideSize - 1)
                result += '\n';
        }
        return result;
    }

    public override void Move(int x, int y, int dx, int dy, Game game)
    {

```

```

        if ((dy < 0 && y == 0) || (dx < 0 && x == 0) || (dx > 0 && x ==
→ game.Board.Count - 1) ||
            (dy > 0 && y == game.Board[0].Count - 1))
        {
            game.Display();
            return;
        }

        var saveItem = game.Board[x + dx][y + dy];
        game.Board[x + dx][y + dy] = this;
        game.Board[x][y] = saveItem;
        game.Process();
        game.Display();
    }

    public BasicItem(char displayChar, ConsoleColor color =
→ ConsoleColor.White): base(displayChar, color)
{
}
}

}

```

9. EmptyItem - пустой игровой элемент

```

namespace SuperGame;

public class EmptyItem: BasicItem
{
    public EmptyItem() : base(' ', ConsoleColor.Black)
    {

    }

    public override string Display(int sideSize)
    {
        return base.Display(sideSize);
    }

    public override void Move(int dx, int dy, int i, int dy1, Game game)
    {
        throw new NotImplementedException();
    }
}

```

10. GameStatus - перечисление всех состояний игры (класса Game)

```

namespace SuperGame;

public enum GameStatus

```

```

    {
        NotChoose,
        Choose,
        Processing,
        Reconfiguring,
        WantEscape
    }
}

```

11. Game - экран, который реализует саму игру

```

using System.Runtime.InteropServices.ComTypes;

namespace SuperGame;

public class Game : IScreen
{
    private GameEngine _gameEngine;

    public GameItem[] GameItems =
    {
        new BasicItem('X', ConsoleColor.Blue), new BasicItem('Y',
        ConsoleColor.Yellow), new BasicItem('@'),
        new BasicItem('@', ConsoleColor.Magenta), new BasicItem('&',
        ConsoleColor.Red)
    };

    private readonly (int Width, int Height) _size = (20, 3);
    private Random _random = new Random();
    public List<List<GameItem>> Board;
    int _cWidth, _cHeight;
    private (int x, int y) _choosedItemPos = (0, 0);
    private bool _isFinish;
    private bool _isChoosed = false;
    private GameStatus _gameStatus = GameStatus.NotChoose;
    private int _side;
    private Task _choosedTask;

    public Game(GameEngine gameEngine)
    {
        Board = new List<List<GameItem>>();
        for (int i = 0; i < _size.Width; i++)
        {
            Board.Add(new List<GameItem>());
            for (int j = 0; j < _size.Height; j++)
            {
                Board[i].Add(GameItems[_random.Next(GameItems.Len
        )]);
            }
        }
    }
}

```

```

        }

    }

    _gameEngine = gameEngine;
    Display();
    Process();
}

public void DrawSelecetdItem(bool isSelected = false)
{
    var item = Board[_choosedItemPos.x][_choosedItemPos.y];
    (int x, int y) pos = (_cWidth / 2 - _side * _size.Width / 2 +
    ← _side * _choosedItemPos.x,
     ← _cHeight / 2 - _side * _size.Height / 2 + _side *
    ← _choosedItemPos.y);
    Utils.WriteLineOnPos(pos.x, pos.y,
                        item.Display(_side), item.Color, isSelected ?
    ← ConsoleColor.DarkGray : Console.BackgroundColor);
}

public async Task DrawChoosedItem()
{
    while (_gameStatus == GameStatus.Choosen)
    {
        DrawSelecetdItem();
        await Task.Delay(100);
        DrawSelecetdItem(true);
        await Task.Delay(100);
    }
}

public void CheckInput()
{
    while (_gameEngine.GameEngineStatus == GameEngineStatus.Playing)
    {
        while (!Console.KeyAvailable)
        {
            if (_isFinish) return;
        }

        if (_gameStatus == GameStatus.NotChoose)
        {
            var key = Console.ReadKey().Key;
            if (Constants.UsedKeys.Contains(key))
            {
                DrawSelecetdItem();
            }
        }
    }
}

```

```

        switch (key)
    {
        case ConsoleKey.S:
        case ConsoleKey.DownArrow:
            if (_choosedItemPos.y ==
→   _size.Height - 1)
                {
                    _choosedItemPos.y = 0;
                }
            else
                {
                    _choosedItemPos.y += 1;
                }

            break;
        case ConsoleKey.W:
        case ConsoleKey.UpArrow:

            if (_choosedItemPos.y == 0)
                {
                    _choosedItemPos.y =
→   _size.Height - 1;
                }
            else
                {
                    _choosedItemPos.y -= 1;
                }

            break;
        case ConsoleKey.D:
        case ConsoleKey.RightArrow:
            if (_choosedItemPos.x ==
→   _size.Width - 1)
                {
                    _choosedItemPos.x = 0;
                }
            else
                {
                    _choosedItemPos.x += 1;
                }

            break;
        case ConsoleKey.A:
        case ConsoleKey.LeftArrow:
            if (_choosedItemPos.x == 0)
                {

```

```

        _choosedItemPos.x =
→   _size.Width - 1;
    }
    else
    {
        _choosedItemPos.x -= 1;
    }

    break;
case ConsoleKey.Enter:
case ConsoleKey.Spacebar:
    _gameStatus = GameStatus.Choosen;
    _choosedTask = DrawChoosedItem();

    break;
case ConsoleKey.Escape:
    Console.Clear();
    string warning = "Нажмите Esc еще
→  раз, если хотите сдаться, если нет, нажмите любую кнопку";
    Utils.WriteLineOnPos(_cWidth / 2
→  - warning.Length / 2, _cHeight / 2, warning);
    _gameStatus =
→  GameStatus.WantEscape;
    break;
}

if (_gameStatus != GameStatus.WantEscape)
    DrawSelectedItem(true);
}

else if (_gameStatus == GameStatus.Choosen)
{
    var key = Console.ReadKey().Key;
    switch (key)
    {
        // Можно перевести игру (Game) в static
        case ConsoleKey.S:
        case ConsoleKey.DownArrow:
            Board[_choosedItemPos.x][_choosed_
→  ItemPos.y].Move(_choosedItemPos.x, _choosedItemPos.y, 0, 1,
→  this);
            break;
        case ConsoleKey.W:
        case ConsoleKey.UpArrow:
            Board[_choosedItemPos.x][_choosed_
→  ItemPos.y].Move(_choosedItemPos.x, _choosedItemPos.y, 0, -1,
→  this);
            break;
        case ConsoleKey.D:

```

```

                case ConsoleKey.RightArrow:
                    Board[_choosedItemPos.x][_choosed]
                    ItemPos.y].Move(_choosedItemPos.x, _choosedItemPos.y, 1, 0,
                    this);
                    break;
                case ConsoleKey.A:
                case ConsoleKey.LeftArrow:
                    Board[_choosedItemPos.x][_choosed]
                    ItemPos.y].Move(_choosedItemPos.x, _choosedItemPos.y, -1, 0,
                    this);
                    break;
            }
        }

    }

    else if (_gameStatus == GameStatus.WantEscape)
    {
        var key = Console.ReadKey().Key;
        switch (key)
        {
            case ConsoleKey.Escape:
                this.Finish();
                break;
            default:
                _gameStatus =
        GameStatus.NotChoose;
                Display();
                break;
        }
    }

    //Display();
}

}

public void CheckEmpty()
{
    for (int i = _size.Width - 1; i >= 0; i--)
    {
        for (int j = _size.Height - 1; j >= 0; j--)
        {
            if (Board[i][j] is EmptyItem)
            {
                Board[i][j] =
        GameItems[_random.Next(GameItems.Length)];
            }
        }
    }
}
}

```

```

public void Process()
{
    _gameStatus = GameStatus.Processing;
    bool isNotChanged = false;
    for (int j = _size.Height - 1; j >= 0; j--)
    {
        for (int i = _size.Width - 1; i >= 0; i--)
        {
            (int h, int v) = (i, j);
            var item = Board[i][j];
            if (item is EmptyItem)
            {
                continue;
            }

            int howH = 1;
            int howV = 1;
            h--;
            // По хорошему надо переопределить сравнение item.
            while (h >= 0 && Board[h][j].DisplayChar ==
→ item.DisplayChar && Board[h][j].Color == item.Color)
            {
                howH++;
                h--;
            }

            v--;
            while (v >= 0 && Board[i][v].DisplayChar ==
→ item.DisplayChar && Board[i][v].Color == item.Color)
            {
                howV++;
                v--;
            }

            if (howV >= 3)
            {
                for (int hj = 0; hj < howV; hj++)
                {
                    Board[i][j - hj] = new
→ EmptyItem();
                }
            }

            goto reprocess;
        }
    }

    if (howH >= 3)
    {

```

```

                for (int hi = 0; hi < howH; hi++)
                {
                    Board[i - hi][j] = new
        ↵ EmptyItem();
                }

                goto reprocess;
            }
        }
    }

    _gameStatus = GameStatus.NotChoose;
    isNotChanged = true;
    reprocess:
    if (!isNotChanged)
    {
        Display();
        CheckEmpty();
        Thread.Sleep(500);
        Display();
        Thread.Sleep(500);
        Process();
    }
}

public void Display()
{
    Console.Clear();

    (_cWidth, _cHeight) = (Console.WindowWidth, Console.WindowHeight);
    _side = Math.Min(_cWidth, _cHeight) / Math.Min(_size.Height,
        ↵ _size.Width);
    while (_side * _size.Width > _cWidth - 1 || _side * _size.Height
        ↵ > _cHeight - 1)
    {
        _side--;
    }

    (int x, int y) pos = (_cWidth / 2 - _side * _size.Width / 2,
        ↵ _cHeight / 2 - _side * _size.Height / 2);

    for (int i = 0; i < _size.Width; i++)
    {
        pos.y = _cHeight / 2 - _side * _size.Height / 2;
        for (int j = 0; j < _size.Height; j++)
        {

```

```

                if (i == _choosedItemPos.x && j ==
→ _choosedItemPos.y)
{
    Utils.WriteLineOnPos(pos.x, pos.y,
                        Board[i][j].Display(_side),
→ Board[i][j].Color, ConsoleColor.DarkGray);
}
else
{
    Utils.WriteLineOnPos(pos.x, pos.y,
                        Board[i][j].Display(_side),
→ Board[i][j].Color);
}

pos.y += _side;
}

pos.x += _side;
}

public void Finish()
{
    _isFinish = true;
    _gameEngine.GameEngineStatus = GameEngineStatus.NotStarted;
}

public void OnReConfigure()
{
    throw new NotImplementedException();
}

public void Start()
{
    var _inputTask = Task.Run(() => CheckInput());
    _inputTask.Wait();
}
}

```

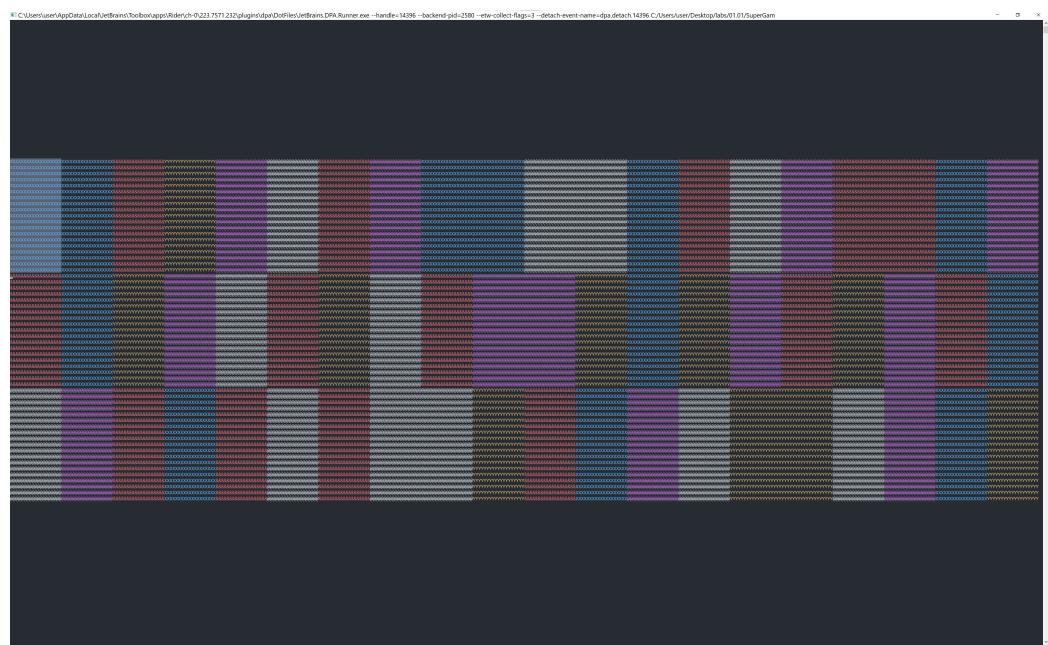
Демонстрация игры:

- Меню

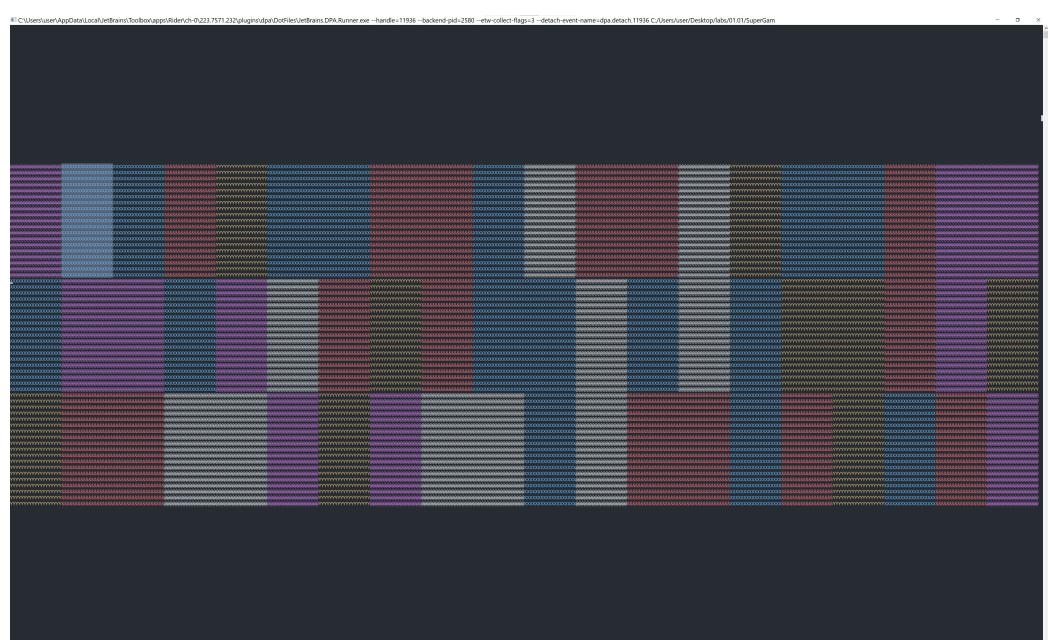
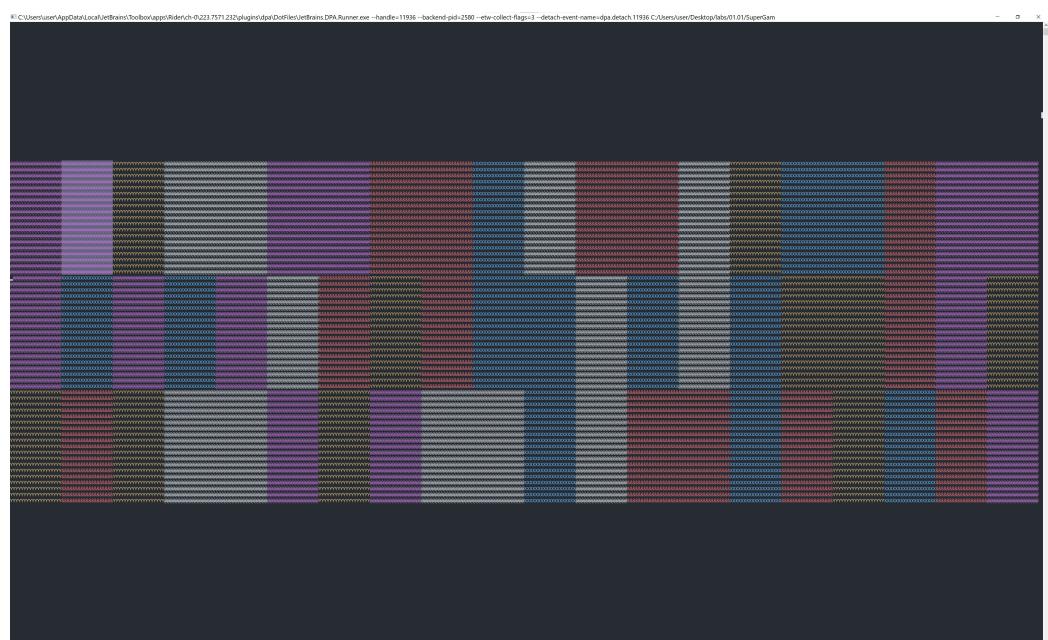


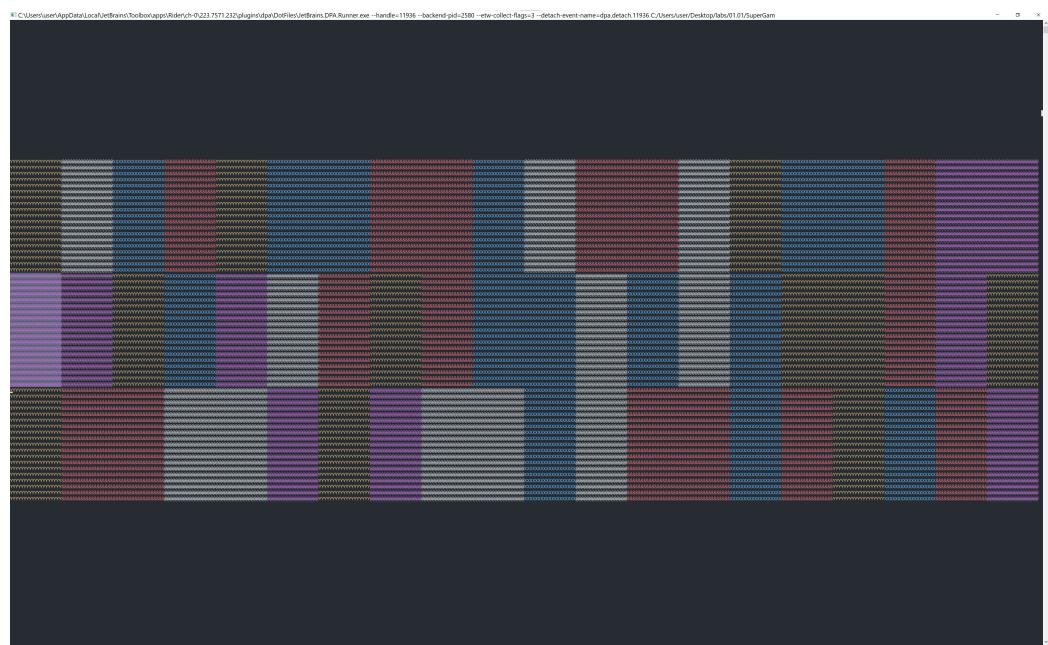
- Демонстрация изменения интерфейса от размера окна



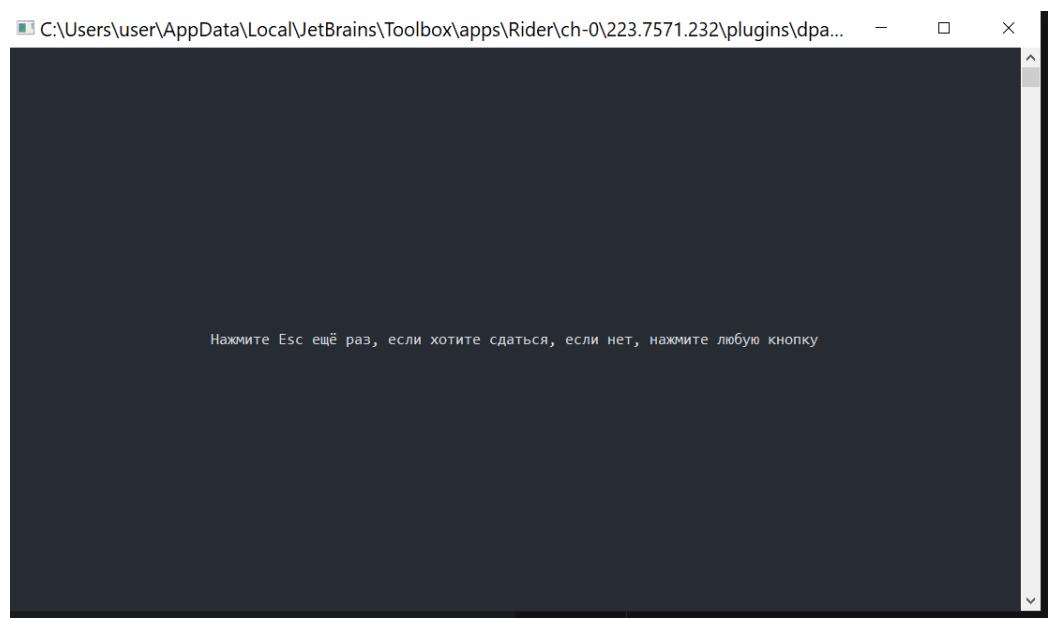


- Делаю несколько ходов:





- Из игры можно выйти в любой момент:



Вывод: я научился работать с абстрактными классами и с консолью на языке C#.

5 Лабораторная работа №5

Цель работы: получение практических навыков при создании классов и запуске приложений в командной строке.

1. Выполнить пример, разобранный на лекции(Классы Person и Employee).

- Person.java

```
class Person {  
    private String name;  
  
    public String getName() {  
        return name;  
    }  
  
    public Person(String name) {  
        this.name = name;  
    }  
  
    public void display() {  
        System.out.println(this.name);  
    }  
}
```

- Employee.java

```
class Employee extends Person {  
  
    private String company;  
  
    public Employee(String name, String company) {  
  
        super(name);  
        this.company = company;  
    }  
  
    @Override  
    public void display() {  
        System.out.println("Name: " + getName());  
        System.out.println("Works in " + company);  
    }  
}
```

- Program.java

```
class Program{  
    public static void main(String[] args){  
        Person person1 = new Person("Vasya Pankov");  
        person1.display();  
        Employee employee = new Employee("Кирилл Гайкин", "pank.su");  
        employee.display();  
    }  
}
```

Результат выполнения программы:

```
C:\Users\user\Desktop\labs_sem_6\01.01\JavaFirst\first>javac Program.java
C:\Users\user\Desktop\labs_sem_6\01.01\JavaFirst\first>java Program
Vasya Pankov
Name: Кирилл Гайкин
Works in pank.su
```

2. Создать приложение, в котором для товара стоимостью а руб. б коп. при оплате за него с руб. д коп. вычисляется, сколько сдачи требуется получить.

- Main.java (Основной класс программы)

```
package su.pank;

import java.util.Scanner;

/**
 * Приложение, в котором для товара стоимостью а руб. б коп. при оплате за
 * него с руб. д коп.
 * вычисляется, сколько сдачи требуется получить.
 *
 * @author Pankov Vasya (pank-su)
 *
 */
public class Main {
    public static void main(String[] args) throws Exception {
        int a, c;
        byte b, d;
        Scanner sc = new Scanner(System.in);
        System.out.println("Введите цену товара (сначала рубли,
        потом копейки)");
        a = sc.nextInt();
        b = sc.nextByte();
        if (b >= 100) {
            a += b / 100;
            b %= 100;
        }
        System.out.println("Введите сумму оплаты (сначала рубли,
        потом копейки)");
        c = sc.nextInt();
        if (c < a) throw new Exception("Суммы недостаточно для
        оплаты");
        d = sc.nextByte();
        if (d > 100) {
            c += d / 100;
            d %= 100;
        }
        if (c == a && d < b) throw new Exception("Суммы
        недостаточно для оплаты");
    }
}
```

```

        boolean minus_one = b > d;
        System.out.printf("Ваша сдача: %d руб. %d к.", (minus_one ?
        ↵    c - a - 1 : c - a), (minus_one ? 100 + d - b : d - b));

    }
}

```

Результаты выполнения программы:

```

Введите цену товара (сначала рубли, потом копейки)
10 1
Введите сумму оплаты (сначала рубли, потом копейки)
11 0
Ваша сдача: 0 руб. 99 к.

```

Рисунок 1 – Обычная работа программы

```

> Task :Main.main()
Введите цену товара (сначала рубли, потом копейки)
10 1
Введите сумму оплаты (сначала рубли, потом копейки)
9
Exception in thread "main" java.lang.Exception Create breakpoint : Суммы недостаточно для оплаты

```

Рисунок 2 – При вводе суммы меньшей чем цена товара

```

Введите цену товара (сначала рубли, потом копейки)
10 1
Введите сумму оплаты (сначала рубли, потом копейки)
10 0
Exception in thread "main" java.lang.Exception Create breakpoint : Суммы недостаточно для оплаты
at su.pank.Main.main(Main.java:32)

```

Рисунок 3 – При вводе суммы меньшей чем цена товара

```
Введите цену товара (сначала рубли, потом копейки)
10 101
Введите сумму оплаты (сначала рубли, потом копейки)
11 1
Ваша сдача: 0 руб. 0 к.
```

Рисунок 4 – Конвертация копеек

3. Создать собственный класс(классы) в соответствии с вариантом, полученным в лабораторной работе по С#(Создание классов).

- SquareMatrix.java (Основной класс матрицы)

```
package org.example;

/** Класс квадратная матрица, поля класса - размерность и элементы матрицы.
 * Методы класса: проверка, является ли матрица верхнетреугольной или
 * низнетреугольной, вывод матрицы.
 * В классе предусмотреть методы: сложение, вычитание, умножение матриц,
 * умножение матрицы на число.
 *
 * @author Vasya Pankov
 * @author ChatGPT(help) */
public class SquareMatrix {
    private int _n;

    public int get_n() {
        return _n;
    }

    private int[][] _matrix;

    public int[][] get_matrix() {
        return _matrix;
    }

    public void set_matrix(int[][] _matrix) throws Exception {
        if (_matrix.length != _n || _matrix[0].length != _n)
            throw new Exception("Вводимая матрица неподходит по
размерности");
        this._matrix = _matrix;
    }

    public void set_n(int _n) throws Exception {
        if (_n <= 0) {
            throw new Exception("Размерность матрицы должна
быть больше 0");
        }
        this._n = _n;
    }

    public SquareMatrix(int n) throws Exception {
        set_n(n);
        set_matrix(new int[n][n]);
    }
}
```

```

public SquareMatrix(int n, int[][] matrixList) throws Exception {
    set_n(n);
    set_matrix(matrixList.clone());
}

@Override
public String toString() {
    StringBuilder output = new StringBuilder("[\n");
    for (int[] list : get_matrix()) {
        output.append('\t');
        for (int el : list) {
            output.append(el + "\t");
        }
        output.append('\n');
    }

    output.append("]");
    return output.toString();
}

public boolean IsUpperTriangular() {
    for (int i = 0; i < _n; i++) {
        for (int j = 0; j < i; j++) {
            if (_matrix[i][j] != 0)
                return false;
        }
    }

    return true;
}

// Это нижняя треугольная матрица?
public boolean IsBottomTriangular() {
    for (int i = 0; i < _n; i++) {
        for (int j = _n - 1; j > i; j--) {
            if (_matrix[i][j] != 0)
                return false;
        }
    }

    return true;
}

public SquareMatrix add(SquareMatrix squareMatrix2) throws
→ Exception {
    if (squareMatrix2._n != this._n)
        throw new RuntimeException("Матрицы не совпадают по
→ размерности");
}

```

```

        SquareMatrix newMatrix = new SquareMatrix(this._n);
        for (int i = 0; i < this._n; i++) {
            for (int j = 0; j < this._n; j++) {
                newMatrix._matrix[i][j] =
→      this._matrix[i][j] + squareMatrix2._matrix[i][j];
            }
        }

        return newMatrix;
    }

    public SquareMatrix subtract(SquareMatrix squareMatrix2) throws
→  Exception {
    if (squareMatrix2._n != this._n)
        throw new RuntimeException("Матрицы не совпадают по
→  размерности");
    SquareMatrix newMatrix = new SquareMatrix(this._n);
    for (int i = 0; i < this._n; i++) {
        for (int j = 0; j < this._n; j++) {
            newMatrix._matrix[i][j] =
→      this._matrix[i][j] - squareMatrix2._matrix[i][j];
        }
    }

    return newMatrix;
}

public SquareMatrix multiply(int n) throws Exception {
    SquareMatrix newMatrix = new SquareMatrix(this._n);
    for (int i = 0; i < this._n; i++) {
        for (int j = 0; j < this._n; j++) {
            newMatrix._matrix[i][j] =
→      this._matrix[i][j] * n;
        }
    }

    return newMatrix;
}

public SquareMatrix multiply(SquareMatrix squareMatrix2) throws
→  Exception {
    if (squareMatrix2._n != this._n)
        throw new RuntimeException("Матрицы не совпадают по
→  размерности");
    SquareMatrix newMatrix = new SquareMatrix(this._n);
    for (int i = 0; i < this._n; i++) {
        for (int j = 0; j < this._n; j++) {

```

```

        int cellValue = 0;
        for (int j1 = 0; j1 < this._n; j1++) {
            cellValue += this._matrix[i][j1] *
→    squareMatrix2._matrix[j1][j];
        }

        newMatrix._matrix[i][j] = cellValue;
    }

    return newMatrix;
}
}

```

- SquareMatrixTests (Класс для тестирования)

```

import static org.junit.jupiter.api.Assertions.assertArrayEquals;
import static org.junit.jupiter.api.Assertions.assertEquals;

import org.example.SquareMatrix;
import org.junit.jupiter.api.DisplayName;
import org.junit.jupiter.api.Nested;
import org.junit.jupiter.api.Test;

/**
 *
 * @author Pankov Vasya
 * @author ChatGPT (help with math tests)
 */
@DisplayName("Тестирование матрицы")
public class SquareMatrixTests {
    @Test
    @DisplayName("Проверка инициализации")
    public void InitTest() throws Exception {
        SquareMatrix matrix = new SquareMatrix(2);
        int[][] myMatrix = new int[2][2];
        myMatrix[0][0] = 0;
        myMatrix[0][1] = 0;
        myMatrix[1][1] = 0;
        myMatrix[1][0] = 0;
        assertArrayEquals(matrix.get_matrix(), myMatrix);
        myMatrix[0][1] = 1;
        matrix = new SquareMatrix(2, myMatrix);
        assertArrayEquals(matrix.get_matrix(), myMatrix);
    }

    @Test
    @DisplayName("Проверка вывода в строку")
    public void StringTest() throws Exception {

```

```

        assertEquals((new SquareMatrix(1)).toString(),
→      "[\n\t0\n]");

}

@Test
@DisplayName("Проверка, верхнетреугольной или нижнетреугольной ли у
→ нас матрица?")
public void TriangularTest() throws Exception {
    int[][] myMatrix = {
        {1, 0},
        {1, 1}
    };
    SquareMatrix matrix = new SquareMatrix(2, myMatrix);
    assert matrix.IsBottomTriangular();
    assert !matrix.IsUpperTriangular();
}

@Nested
@DisplayName("Математические тесты")
public class MathTests{
    @Test
    @DisplayName("Сложение двух матриц")
    public void testAddition() throws Exception {
        int[][] a = {{1, 2}, {3, 4}};
        int[][] b = {{5, 6}, {7, 8}};
        SquareMatrix matrixA = new SquareMatrix(2, a);
        SquareMatrix matrixB = new SquareMatrix(2, b);
        SquareMatrix result = matrixA.add(matrixB);
        int[][] expected = {{6, 8}, {10, 12}};
        assertEquals(expected, result.get_matrix());
    }

    @Test
    @DisplayName("Вычитание двух матриц")
    public void testSubtraction() throws Exception {
        int[][] a = {{1, 2}, {3, 4}};
        int[][] b = {{5, 6}, {7, 8}};
        SquareMatrix matrixA = new SquareMatrix(2, a);
        SquareMatrix matrixB = new SquareMatrix(2, b);
        SquareMatrix result = matrixA.subtract(matrixB);
        int[][] expected = {{-4, -4}, {-4, -4}};
        assertEquals(expected, result.get_matrix());
    }

    @Test
    @DisplayName("Скалярное умножение матрицы")
    public void testScalarMultiplication() throws Exception {
        int[][] a = {{1, 2}, {3, 4}};

```

```

        SquareMatrix matrixA = new SquareMatrix(2, a);
        SquareMatrix result = matrixA.multiply(2);
        int[][] expected = {{2, 4}, {6, 8}};
        assertEquals(expected, result.get_matrix());
    }

    @Test
    @DisplayName("Умножение матриц")
    public void testMatrixMultiplication() throws Exception {
        int[][] a = {{1, 2}, {3, 4}};
        int[][] b = {{5, 6}, {7, 8}};
        SquareMatrix matrixA = new SquareMatrix(2, a);
        SquareMatrix matrixB = new SquareMatrix(2, b);
        SquareMatrix result = matrixA.multiply(matrixB);
        int[][] expected = {{19, 22}, {43, 50}};
        assertEquals(expected, result.get_matrix());
    }
}

```

Результат выполнения программы:

```

> Task :compileJava
> Task :processResources NO-SOURCE
> Task :classes
> Task :compileTestJava UP-TO-DATE
> Task :processTestResources NO-SOURCE
> Task :testClasses UP-TO-DATE
> Task :test
BUILD SUCCESSFUL in 4s
3 actionable tasks: 2 executed, 1 up-to-date
2:17:58 PM: Execution finished ':test --tests "SquareMatrixTests"'.

```

4. Выполнить компиляцию и запуск приложения Java с помощью командной строки.

- Test.java

```

import java.util.Scanner;

public class Test {
    public static void main(String[] args){
        Scanner scanner = new Scanner(System.in);
        System.out.println("Введите своё имя:");
        String name = scanner.nextLine();
        System.out.println("Введите число:");
        int number = scanner.nextInt();
    }
}

```

```
        System.out.println("Спасибо " + name + "! Вы ввели число "
    ↪    + number);
}
}
```

Компиляция и запуск:

```
C:\Users\user\Desktop\labs_sem_6\01.01\JavaFirst\third>"C:\Users\user\.jdks\openjdk-18.0.2.1\bin\javac" Test.java
C:\Users\user\Desktop\labs_sem_6\01.01\JavaFirst\third>dir
Volume in drive C has no label.
Volume Serial Number is 2C8D-D5B5

Directory of C:\Users\user\Desktop\labs_sem_6\01.01\JavaFirst\third

02/20/2023  01:52 PM    <DIR>      .
02/20/2023  01:52 PM    <DIR>      ..
02/20/2023  02:20 PM           1,189 Test.class
02/20/2023  01:52 PM           419 Test.java
02/20/2023  01:45 PM           0 Test.java~
                           3 File(s)       1,608 bytes
                           2 Dir(s)  92,801,454,080 bytes free

C:\Users\user\Desktop\labs_sem_6\01.01\JavaFirst\third>"C:\Users\user\.jdks\openjdk-18.0.2.1\bin\java" Test
Введите своё имя:
Vasya
Введите число:
12
Спасибо Vasya! Вы ввели число 12
```

Рисунок 5 – Демонстрация компиляции и запуска приложения на Java