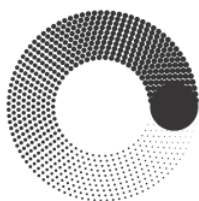


**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ  
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**



**МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**Факультет информационных технологий  
Кафедра Информатики и информационных технологий**

**направление подготовки 09.03.02 «Информационные системы и  
технологии»,**

**ЛАБОРАТОРНАЯ РАБОТА №5**

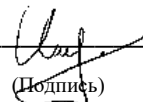
**Дисциплина:** Технология прикладного программирования

**Выполнил:** студент группы 231-338

Шаура И. М.

**Дата, подпись** 10.02.2024

(Дата)

  
(Подпись)

**Проверил:** ст. преп. Калмыков Е. А.

(Оценка)

**Дата, подпись** \_\_\_\_\_

(Дата)

(Подпись)

**Замечания:**

---

---

**Москва**

**2024**

## Задание:

Разработать программу "блокнот" с соответствующим функционалом.

```
<Window x:Class="_5.Notepad.MainWindow"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:local="clr-namespace:_5.Notepad"
    mc:Ignorable="d"
    Title="{Binding FileNameWithAsterisk, UpdateSourceTrigger=PropertyChanged}" Height="450"
    Width="800" Closing="Window_Closing">
    <Window.InputBindings>
        <KeyBinding Command="{Binding SaveCommand}" Modifiers="Ctrl" Key="S" />
        <KeyBinding Command="{Binding NewFileCommand}" Modifiers="Ctrl" Key="N" />
        <KeyBinding Command="{Binding OpenFileCommand}" Modifiers="Ctrl" Key="O" />
    </Window.InputBindings>

    <DockPanel PreviewMouseWheel="DockPanel_PreviewMouseWheel">
        <Menu VerticalAlignment="Top" DockPanel.Dock="Top">
            <MenuItem Header="Файл">
                <MenuItem Header="Новый файл" Click="MenuItem_ClickNewFile"/>
                <MenuItem Header="Открыть файл" Click="MenuItem_ClickOpenFile"/>
                <MenuItem Header="Сохранить" Click="MenuItem_ClickSave"/>
                <Separator />
                <MenuItem Header="Выход" Click="MenuItem_ClickExit"/>
            </MenuItem>
        </Menu>
        <TextBox FontFamily="Consolas" FontSize="{Binding CurrentFontSize,
            UpdateSourceTrigger=PropertyChanged}" Text="{Binding MainText,
            UpdateSourceTrigger=PropertyChanged}" AcceptsReturn="True" HorizontalScrollBarVisibility="Auto"
            VerticalScrollBarVisibility="Auto"/>
    </DockPanel>
</Window>
```

```
using Microsoft.Win32;
using System.ComponentModel;
using System.IO;
using System.Windows;
using System.Windows.Input;

namespace _5.Notepad
{
    public class ActionCommand(Action action) : ICommand
    {
        private readonly Action _action = action;

        public void Execute(object parameter)
        {
            _action();
        }

        public bool CanExecute(object parameter)
        {
            return true;
        }

        public event EventHandler CanExecuteChanged;
    }
}
```

```

}
public partial class MainWindow : Window, INotifyPropertyChanged
{
    public event PropertyChangedEventHandler? PropertyChanged;
    private void OnPropertyChanged(string propertyName)
    {
        PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propertyName));
    }

    private int _currentFontSize = 14;

    public int CurrentFontSize
    {
        get { return _currentFontSize; }
        set
        {
            _currentFontSize = value;
            OnPropertyChanged(nameof(CurrentFontSize));
        }
    }

    private ICommand _saveCommand;
    public ICommand SaveCommand => _saveCommand ??= new ActionCommand(SaveText);

    private ICommand _newFileCommand;
    public ICommand NewFileCommand => _newFileCommand ??= new ActionCommand(NewFile);

    private ICommand _openFileCommand;
    public ICommand OpenFileCommand => _openFileCommand ??= new ActionCommand(OpenFile);

    private bool _isSaved = false;

    private string _path = string.Empty;

    private string Asterisk => FileHasChanges ? "*" : "";

    public string FileNameWithAsterisk
    {
        get => Asterisk + FileName;
        set
        {
            OnPropertyChanged(nameof(FileNameWithAsterisk));
            FileHasChanges = true;
        }
    }

    private string _fileName = "Безымянный.txt";
    public string FileName
    {
        get => _fileName;
        set
        {
            _fileName = value;
            OnPropertyChanged(nameof(FileName));
            OnPropertyChanged(nameof(FileNameWithAsterisk));
            FileHasChanges = true;
        }
    }

    private bool _fileHasChanges = true;
    public bool FileHasChanges

```

```

{
    get => _fileHasChanges;
    set
    {
        _fileHasChanges = value;
        OnPropertyChanged(nameof(FileHasChanges));
        OnPropertyChanged(nameof(FileNameWithAsterisk));
    }
}

private string _mainText;

public string MainText
{
    get => _mainText;
    set
    {
        _mainText = value;
        OnPropertyChanged(nameof(MainText));
        FileHasChanges = true;
    }
}

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

private void SaveText()
{
    if (!_isSaved)
    {
        File.WriteAllText(_path, MainText);
        FileHasChanges = false;
        return;
    }

    SaveFileDialog saveFileDialog = new()
    {
        Title = "Сохранение",
        Filter = "Text files (*.txt)|*.txt|All files (*.*)|*.*",
        FilterIndex = 2,
        RestoreDirectory = true,
        FileName = FileName
    };

    if (saveFileDialog.ShowDialog() == true)
    {
        _path = saveFileDialog.FileName;
        _isSaved = true;
        FileName = saveFileDialog.SafeFileName;
        FileHasChanges = false;

        File.WriteAllText(_path, MainText);
    }
}

private void SaveTextMessageBox()
{
    if (!FileHasChanges) return;
    switch (MessageBox.Show(
        $"Вы хотите сохранить изменения в файле \"{FileName}\"",

```

```

        "Сохранить",
        MessageBoxButton.YesNoCancel,
        MessageBoxImage.Question,
        MessageBoxResult.Yes))
    {
        case MessageBoxResult.None:
            break;
        case MessageBoxResult.OK:
            break;
        case MessageBoxResult.Cancel:
            return;
        case MessageBoxResult.Yes:
            SaveText();
            break;
        case MessageBoxResult.No:
            break;
        default:
            break;
    }
}
private void NewFile()
{
    SaveTextMessageBox();
    _isSaved = false;

    MainText = string.Empty;
}
private void OpenFile()
{
    SaveTextMessageBox();

    OpenFileDialog openFileDialog = new()
    {
        Title = "Открыть",
        Filter = "Text files (*.txt)|*.txt",
        FilterIndex = 1,
        RestoreDirectory = true
    };

    if (openFileDialog.ShowDialog() == true)
    {
        _isSaved = true;
        _path = openFileDialog.FileName;
        FileName = openFileDialog.SafeFileName;
        FileHasChanges = false;
        MainText = File.ReadAllText(_path);
    }
}

private void MenuItem_ClickNewFile(object sender, RoutedEventArgs e)
{
    NewFile();
}
private void MenuItem_ClickOpenFile(object sender, RoutedEventArgs e)
{
    OpenFile();
}
private void MenuItem_ClickExit(object sender, RoutedEventArgs e)
{
    SaveTextMessageBox();
    Application.Current.Shutdown();
}
private void MenuItem_ClickSave(object sender, RoutedEventArgs e)

```

```
{
    SaveText();
}

private void Window_Closing(object sender, CancelEventArgs e)
{
    SaveTextMessageBox();
}

private void DockPanel_PreviewMouseWheel(object sender, MouseWheelEventArgs e)
{
    if (Keyboard.IsKeyDown(Key.LeftCtrl))
    {
        if (e.Delta > 0)
        {
            if (CurrentFontSize < 50) CurrentFontSize++;
        }

        if (e.Delta < 0)
        {
            if (CurrentFontSize > 1) CurrentFontSize--;
        }
        e.Handled = true;
    }
}
}
```

