Министерство науки и высшего образования

Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего профессионального образования

«Российский государственный университет им. А.Н. Косыгина

(Технологии.Дизайн.Искусство)»

Кафедра автоматизированных систем обработки информации и управления

Лабораторная работа №2

Выполнила: Букша Кирилл Владимирович

Группа: МАГ-В-221

Вариант 4

Проверил: Кузьмина Тамара Михайловна

Москва 2021

Задание:

Изучить примеры реализации событий на языке C#. Реализовать панель для ввода данных и вывода накопленной информации о событиях.

Вариант: Автор – издательства. Событии – написание произведения, о нем сообщается название, жанр, количество страниц. В программе должны создаваться один объект класса Автор и два объекта класса Издательство.

Решение:

Были написаны классы. Класс «Автор»:

```
namespace lab2
{
    public class Author
    {
        public Author(string name)
        {
            Name = name;
        }

        public string Name { get; }

        public event App.ReceiveAuthorWork PublishEvent;

        public void Publish(AuthorWork authorWork)
        {
            PublishEvent?.Invoke(this, authorWork);
        }
    }
}
```

Класс «Издательство»:

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

namespace lab2
{
    public class Agency
    {
        public Agency(string name)
        {
            Name = name;
            Received = new List<AuthorWork>();
            SubscribedAuthors = new List<Author>();
        }

        public string Name { get; }

        public List<AuthorWork> Received { get; }

        private List<Author> SubscribedAuthors { get; }

        public void ReceiveAuthorWork(object sender, AuthorWork authorWork)
        {
            Received.Add(authorWork);
        }
```

```
        public void Subscribe(Author author)
        {
            if (author == null) throw new ArgumentNullException(nameof(author));
            if (!SubscribedAuthors.Contains(author))
            {
                SubscribedAuthors.Add(author);
                author.PublishEvent += ReceiveAuthorWork;
            }
        }

        public void Unsubscribe(Author author)
        {
            if (author == null) throw new ArgumentNullException(nameof(author));
            SubscribedAuthors.Remove(author);
            author.PublishEvent -= ReceiveAuthorWork;
        }
    }
}
```

Для корректной работы, в основном классе программы размещается делегат:

```
namespace lab2
{
    /// <summary>
    ///     Interaction logic for App.xaml
    /// </summary>
    public class App : Application
    {
        public delegate void ReceiveAuthorWork(object sender, AuthorWork authorWork);
    }
}
```

Также был разработан класс, представляющий авторскую работу:

```
namespace lab2
{
    public class AuthorWork
    {
        public AuthorWork(string genre, string title, int pagesCount)
        {
            Genre = genre;
            Title = title;
            PagesCount = pagesCount;
        }

        public string Genre { get; }
        public string Title { get; }
        public int PagesCount { get; }

        public bool ValidateData()
        {
            return !(string.IsNullOrEmpty(Genre) ||
                     string.IsNullOrEmpty(Title) ||
                     PagesCount == 0);
        }

        public override string ToString()
        {
            return $"{Genre}, {Title}, {PagesCount} c.";
        }
    }
}
```

Этот класс содержит все необходимые поля и методы валидации и преобразования к текстовому виду. Последний используется для корректного отображения событий в компоненте ListBox.

Класс, связанный с главной формой содержит внутри себя описание событий, вызываемых по нажатию кнопок и выглядит следующим образом:

```
using System;
using System.Globalization;
using System.Text.RegularExpressions;
using System.Windows;
using System.Windows.Input;

namespace lab2
{
    /// <summary>
    ///     Interaction logic for MainWindow.xaml
    /// </summary>
    public partial class MainWindow : Window
    {
        public Agency agency1 = new("Agency1");
        public Agency agency2 = new("Agency2");
        public Author author = new("Test Author");

        public MainWindow()
        {
            InitializeComponent();
            Agency1ListBox.DataContext = agency1.Received;
            Agency2ListBox.DataContext = agency2.Received;
        }

        private void PublishButton_Click(object sender, RoutedEventArgs e)
        {
            var genre = Genre.Text;
            var title = Title.Text;
            var pagesCountString = PagesCount.Text;
            var pagesCount = 0;
            try
            {
                pagesCount = int.Parse(pagesCountString, NumberStyles.Any);
            }
            catch (Exception exception)
            {
                MessageBox.Show($"Неправильный формат данных в поле
\"{PagesCountLabel.Content}\"!", "Ошибка",
                    MessageBoxButton.OK, MessageBoxImage.Error);
            }

            var authorWork = new AuthorWork(genre, title, pagesCount);
            if (!authorWork.ValidateData())
                MessageBox.Show("Введена недопустимая информация. Проверьте поля!",
"Ошибка",
                    MessageBoxButton.OK, MessageBoxImage.Error);

            author.Publish(authorWork);
            Agency1ListBox.Items.Refresh();
            Agency2ListBox.Items.Refresh();
        }

        private void NumberValidationTextBox(object sender, TextCompositionEventArgs e)
        {
            var regex = new Regex("[^0-9]+");
            e.Handled = regex.IsMatch(e.Text);
```

```
        }

        private void Subscribe1Button_Click(object sender, RoutedEventArgs e)
        {
            agency1.Subscribe(author);
        }

        private void Unsubscribe1Button_Click(object sender, RoutedEventArgs e)
        {
            agency1.Unsubscribe(author);
        }

        private void Subscribe2Button_Click(object sender, RoutedEventArgs e)
        {
            agency2.Subscribe(author);
        }

        private void Unsubscribe2Button_Click(object sender, RoutedEventArgs e)
        {
            agency2.Unsubscribe(author);
        }
    }
}
```

Результаты работы программы изображены на рисунке 1.

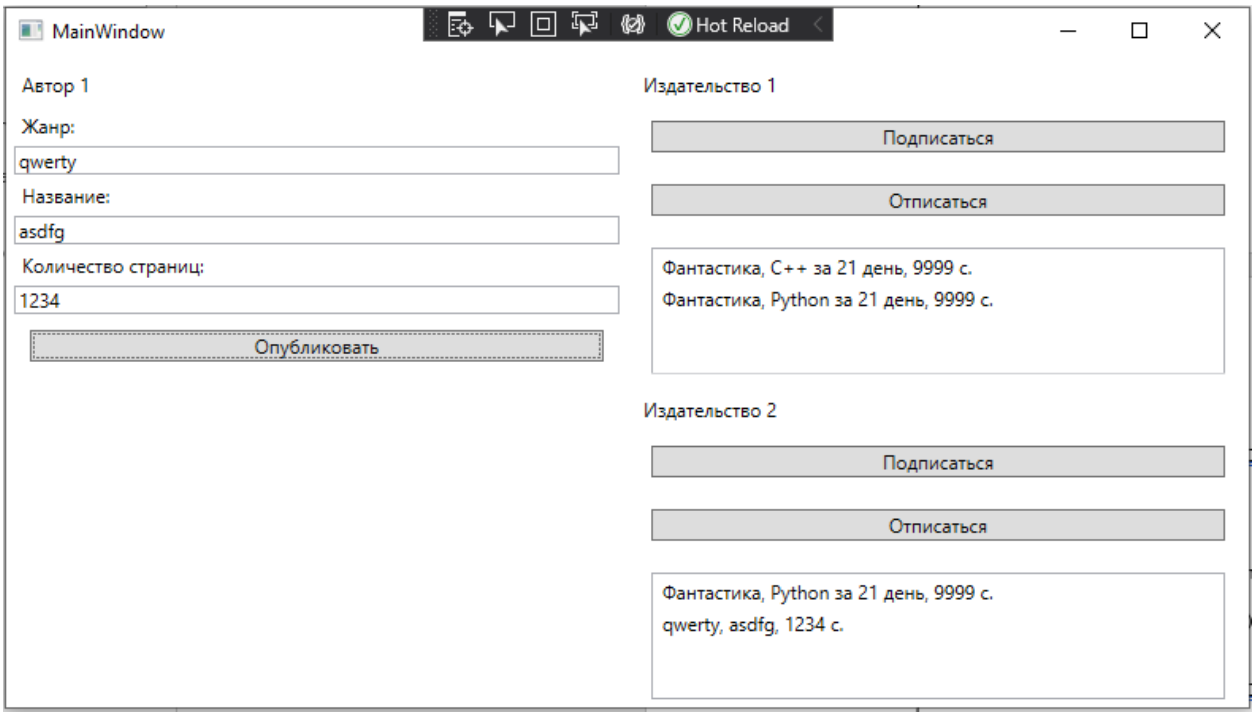

Рисунок 1. Окно программы.

Отредактируем исходные данные таким образом, чтобы валидация входных данных выдавала отрицательный результат:

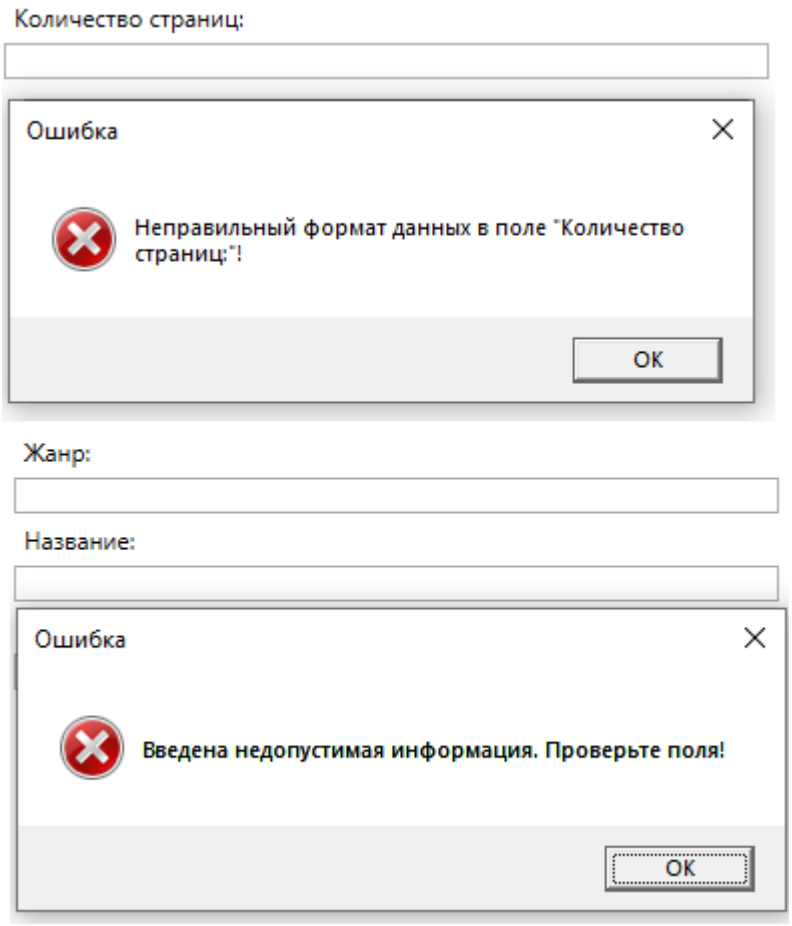

Рисунок 2. Сообщения об ошибках.

Вывод:

Была написана программа, использующая событийный подход к программированию. Программа отлажена и протестирована ручными методами тестирования. Исходный код программы залит на Github и доступен по ссылке: https://github.com/bukSHA1024/RSU_TRPO_Lab2