**13.** **События**

Задание №1

Создать 3 метода, возвращающих три различных сообщения. Создать делегат. В программе вызывать все три метода при помощи делегата. Создать метод, в котором делегат будет параметром.

Листинг программы:

class Program

{

delegate string WorkText(string text);

static event WorkText events;

static void Main()

{

Console.Write("Введите текст: ");

string text = Console.ReadLine();

events += TextToUpper;

events += TextToLower;

events += TextReverse;

WorkEvent(events, text);

}

static string TextToUpper(string text) => text.ToUpper();

static string TextToLower(string text) => text.ToLower();

static string TextReverse(string text)

{

char[] chars = text.ToCharArray();

Array.Reverse(chars);

return new string(chars);

}

static void WorkEvent(WorkText workText, string text) =>

Console.WriteLine(workText?.Invoke(text));

}

Таблица 13.1 – Входные и выходные данные программы задание №1

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| Текст | тскеТ |

Анализ результатов:



Рисунок 13.1 – Результат выполнения программы задание №1

Задание №2

Для класса MyInfo содержащего поле name (Ваше имя) создать событие выводящее оповещение в случае изменения значения поля name.

Листинг программы:

Program

using z3.Model;

Console.Write("Введите имя: ");

string name = Console.ReadLine();

MyInfo myInfo = new MyInfo(name);

Console.Write($"{myInfo.Name} заменить на: ");

myInfo.Name = Console.ReadLine();

MyInfo

namespace z3.Model

{

internal class MyInfo

{

delegate void ChangingInformation();

event ChangingInformation ciEvent;

private string \_name;

public string Name

{

get => \_name;

set

{

if (\_name != value)

{

\_name = value;

ciEvent?.Invoke();

}

}

}

public MyInfo(string name)

{

\_name = name;

ciEvent += delegate ()

{

Console.WriteLine("Имя изменилось");

};

}

}

}

Таблица 13.2 – Входные и выходные данные программы задание №2

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| Антон  Maki | Имя изменилось |

Анализ результатов:

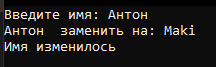


Рисунок 13.2 – Результат выполнения программы задание №2

Задание №3

Листинг программы:

Program

using z4.Model;

Blogger blogger = new Blogger("MakiMails");

Subscriber subscriber = new Subscriber("Alex");

Moderator moderator = new Moderator("123456");

blogger.Attach(subscriber);

blogger.Attach(moderator);

blogger.CreateNewVideo("24/7 пишу код =)");

blogger.Detach(subscriber);

blogger.CreateNewVideo("Пишу код");

IoObserver

using z4.Model;

namespace z4.Interface

{

internal interface IObserver

{

void Update(Video video);

void WhenSubscribing(Blogger blogger);

void WhenUnsubscribing(Blogger blogger);

}

}

Blogger

using z4.Interface;

namespace z4.Model

{

internal class Blogger

{

delegate void UpdateBloger(Video video);

event UpdateBloger? ub;

private string \_name;

public string Name { get => \_name; }

public Blogger(string name)

{

\_name = name;

}

public void Attach(IObserver observer)

{

ub += observer.Update;

observer.WhenSubscribing(this);

}

public void Detach(IObserver observer)

{

ub -= observer.Update;

observer.WhenUnsubscribing(this);

}

public void Notify(Video video)

{

ub?.Invoke(video);

}

public void CreateNewVideo(string nameVideo)

{

Video video = new Video()

{

Author = this,

Name = nameVideo

};

Notify(video);

}

}

}

Video

namespace z4.Model

{

internal class Video

{

public Blogger Author { get; set; }

public string Name { get; set; }

}

}

Subscriber

using System.Security.Cryptography;

using z4.Interface;

namespace z4.Model

{

internal class Subscriber : IObserver

{

private string \_name;

public Subscriber(string name)

{

\_name = name;

}

public void WhenSubscribing(Blogger blogger) => Console.WriteLine($"Подписчик {\_name} подписался на блогера: {blogger.Name} ");

public void WhenUnsubscribing(Blogger blogger) => Console.WriteLine($"Подписчик {\_name} отписался от блогера: {blogger.Name}");

public void Update(Video video)

{

Console.WriteLine($"Подписчик {\_name} был уведомлен об выходе видео " +

$"под названием {video.Name} " +

$"от блогера {video.Author.Name}");

}

}

}

Moderator

using z4.Interface;

namespace z4.Model

{

internal class Moderator : IObserver

{

private string \_id;

public Moderator(string id)

{

\_id = id;

}

public void WhenSubscribing(Blogger blogger) => Console.WriteLine($"Модератор под id: {\_id} прикреплен к блогеру: {blogger.Name} ");

public void WhenUnsubscribing(Blogger blogger) => Console.WriteLine($"Модератор под id: {\_id} откреплен от блогера: {blogger.Name}");

public void Update(Video video)

{

Console.WriteLine($"Модератор под id: {\_id} был уведомлен об выходе видео " +

$"под названием {video.Name} " +

$"от блогера {video.Author.Name}");

}

}

}

Таблица 13.3 – Входные и выходные данные программы задание №3

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
|  |  |

Анализ результатов:

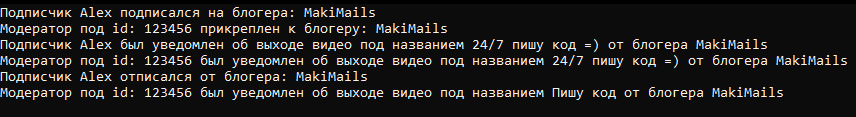


Рисунок 13.3 – Результат выполнения программы задание №3