**13 События**

Задание 1. При решении задач, предусмотреть обработку исключительных ситуаций. Создать 3 метода для подсчета значений трех разных функций. Создать делегат. В программе вызывать все три метода при помощи делегата.

(дополнительно) Задание 2. На основе своей программы создать метод, в котором делегат будет параметром.

Листинг программы:

delegate double MyDelegate(double x); static void Main(string[] args)

{try{MyDelegate myDelegate = Method1; myDelegate += Method2;

myDelegate += Method3;

Console.WriteLine("Результаты методов:");

foreach (MyDelegate method in myDelegate.GetInvocationList())

{try{double result = method(2); Console.WriteLine($"{method.Method.Name}: {result}");}

catch (Exception ex){Console.WriteLine($"Ошибка при выполнении метода {method.Method.Name}: {ex.Message}"); } }

DoSomething(myDelegate, 2); } catch (Exception ex) {

Console.WriteLine($"Ошибка: {ex.Message}"); } }

static double Method1(double x) { return x \* x;}

static double Method2(double x) { return Math.Pow(x,2); }

static double Method3(double x) { return Math.Exp(x); }

static void DoSomething(MyDelegate myDelegate, double x)

{try {foreach (MyDelegate method in myDelegate.GetInvocationList())

{double result = method(x); Console.WriteLine($"Результат метода: {result}");}} catch (Exception ex){ Console.WriteLine($"Ошибка при выполнении метода DoSomething: {ex.Message}");} }

Таблица 13.1 – Выходные и входные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 2 | Результаты методов:  Method1: 4  Method2: 4  Method3: 7,38905609893065  Результат метода: 4  Результат метода: 4  Результат метода: 7,38905609893065 |

Источник: собственная разработка

Анализ результатов:

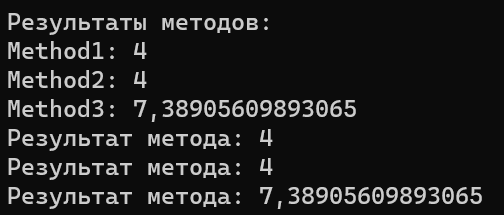


Рисунок 13.1 – Результат работы программы

Источник: собственная разработка

Задание 3. Для класса MyInfo содержащего поле name (Ваше имя) создать событие выводящее оповещение в случае изменения значения поля name.

Листинг программы:

Console.Write("Введите имя: "); string name = Console.ReadLine();

MyInfo myInfo = new MyInfo(name);

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

myInfo.Name = Console.ReadLine();

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 – Выходные и входные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| Артём, Тёма | Имя изменилось |

Источник: собственная разработка

Анализ результатов:

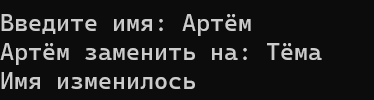


Рисунок 13.2 – Результат работы программы

Источник: собственная разработка

Задание 4. Создать класс, в нем делегат и соответствующее ему событие. Создать два класса-наблюдателя, в методах которых будет описана их реакция на событие. Добавить 2 обработчика к событию из первого класса и один из второго. Вывести результат на экран. Удалить один обработчик события и вывести результат на экран.

Листинг программы:

Program

using \_3.Docum; Photographer photographer = new Photographer("Avstreetc");

Subscriber subscriber = new Subscriber("Steve");

Moderator moderator = new Moderator("1234");

photographer.Attach(subscriber); photographer.Attach(moderator);

photographer.CreateNewPhoto("Сделаю ваш момент в фотографии");

photographer.Detach(subscriber);

photographer.CreateNewPhoto("Фотографирую");

IObserver

using \_3.Docum; namespace \_3.Interfac

{internal interface IObserver{void Update(Photo photo);

void WhenSubscribing(Photographer photographer);

void WhenUnsubscribing(Photographer photographer);}}

Mogerator

using \_3.Interfac; namespace \_3.Docum

{internal class Moderator : IObserver{

private string \_id; public Moderator(string id)

{ \_id = id;}

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

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

public void Update(Photo photo)

{Console.WriteLine($"Модератор под id: {\_id} был уведомлен о новой фотосессии " + $"под названием {photo.Name} " +

$"от фотографа {photo.Author.Name}");}}}

Photo

namespace \_3.Docum{internal class Photo{public Photographer Author{get;set; }

public string Name { get; set; }}}

Photographer

using \_3.Interfac; namespace \_3.Docum

{internal class Photographer { delegate void UpdatePhotograph(Photo photo);

event UpdatePhotograph? ub; private string \_name;

public string Name { get => \_name; } public Photographer(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(Photo photo){ub?.Invoke(photo);}

public void CreateNewPhoto(string namePhoto){ Photo photo = new Photo()

{ Author = this, Name = namePhoto}; Notify(photo);}}}

Subscribers

using \_3.Interfac;namespace \_3.Docum

{internal class Subscriber : IObserver{

private string \_name; public Subscriber(string name)

{ \_name = name;}

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

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

public void Update(Photo photo){

Console.WriteLine($"Подписчик {\_name} был уведомлен об выходе фотосессии " + $"под названием {photo.Name} " +

$"от фотографа {photo.Author.Name}");}}}

Таблица 13.3 – Выходные и входные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
|  | Подписчик Steve подписался на блогера: Avstreetc  Модератор под id: 1234 прикреплен к фотографу: Avstreetc  Подписчик Steve был уведомлен об выходе фотосессии под названием Сделаю ваш момент в фотографии от фотографа Avstreetc  Модератор под id: 1234 был уведомлен о новой фотосессии под названием Сделаю ваш момент в фотографии от фотографа Avstreetc  Подписчик Steve отписался от блогера: Avstreetc  Модератор под id: 1234 был уведомлен о новой фотосессии под названием Фотографирую от фотографа Avstreetc |

Источник: собственная разработка

Анализ результатов:

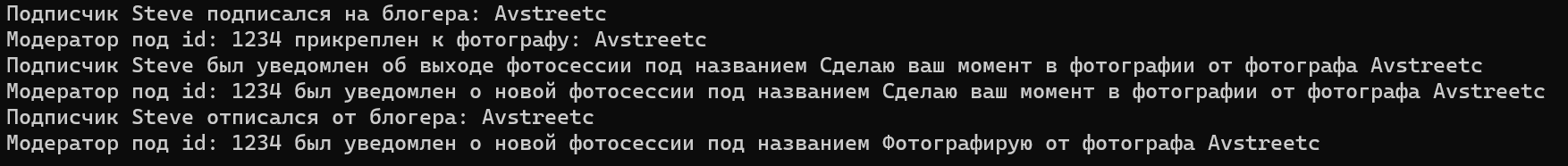


Рисунок 13.3 – Результат работы программы

Источник: собственная разработка