**Обобщенные классы**

**https://metanit.com/sharp/tutorial/3.12.php**

class Person<T>

{

    public T Id { get; set; }

    public string Name { get; set; }

    public Person(T id, string name)

    {

        Id = id;

        Name = name;

    }

}

Person<int> tom = new Person<int>(546, "Tom");

**Делегаты**

**https://metanit.com/sharp/tutorial/3.13.php**

delegate void Message(); // 1. Объявляем делегат

class Program

{

    static void Main()

    {

        Message mes;            // 2. Создаем переменную делегата

        mes = Hello;            // 3. Присваиваем этой переменной адрес метода

        mes();                  // 4. Вызываем метод

        void Hello() => Console.WriteLine("Hello METANIT.COM");

    }

}

Делегаты представляют такие объекты, которые указывают на методы. То есть делегаты - это указатели на методы и с помощью делегатов мы можем вызвать данные методы

**События**

**<https://metanit.com/sharp/tutorial/3.14.php>**

**<https://www.bestprog.net/ru/2017/11/24/events-and-delegates-the-concept-of-event-interaction-between-events_ru/>**

**События** сигнализируют системе о том, что произошло определенное действие. И если нам надо отследить эти действия, то как раз мы можем применять события.

class Account

{

    public delegate void AccountHandler(string message);

    public event AccountHandler? Notify;              // 1.Определение события

    public Account(int sum) => Sum = sum;

    public int Sum { get; private set; }

    public void Put(int sum)

    {

        Sum += sum;

        Notify?.Invoke($"На счет поступило: {sum}");   // 2.Вызов события

    }

    public void Take(int sum)

    {

        if (Sum >= sum)

        {

            Sum -= sum;

            Notify?.Invoke($"Со счета снято: {sum}");   // 2.Вызов события

        }

        else

        {

            Notify?.Invoke($"Недостаточно денег на счете. Текущий баланс: {Sum}"); ;

        }

    }

}

namespace TrainEvents03

{

public partial class Form1 : Form

{

// 1. Объявить тип делегата

delegate void CalcFigure(double R);

// 2. Объявить событие с именем ECF

event CalcFigure ECF;

// 3. Методы обработки события - размещаются в этом же классе // Методы имеют точно такую же сигнатуру как тип делегата CalcFigure // Длина окружности на основе радиуса R

void GetLength(double R)

{

double res;

const double Pi = 3.1415;

res = 2 \* Pi \* R;

label1.Text = "Длина окружности = " + res.ToString(); // вывести на форму

}

// Площадь круга

void GetArea(double R)

{

double res;

const double Pi = 3.1415;

res = Pi \* R \* R;

label2.Text = "Площадь круга = " + res.ToString(); // результат - на форму

}

// Объем шара

void GetVolume(double R)

{

double res;

const double Pi = 3.1415;

res = 4.0 / 3.0 \* Pi \* R \* R \* R;

label3.Text = "Объем шара = " + res.ToString();

}

public Form1()

{

InitializeComponent();

}

private void button1\_Click(object sender, EventArgs e)

{

// 4. Демонстрация работы с методами с помощью события // 4.1. Создать цепочку методов, которые будут вызываться из события ECF

ECF = GetLength; // ECF => GetLength()

ECF += GetArea; // ECF => GetLength() -> GetArea()

ECF += GetVolume; // ECF => GetLength() -> GetArea() -> GetVolume()

// 4.2. Вызов события ECF с параметром 2.0

ECF(2.0); // вызываются последовательно три метода GetLength(), GetArea(), GetVolume()

// Вызов события для параметра 3.5

ECF(3.5);

}

}

}

**LINQ**

**https://metanit.com/sharp/tutorial/15.1.php**

представляет простой и удобный язык запросов к источнику данных. В качестве источника данных может выступать объект, реализующий интерфейс IEnumerable (например, стандартные коллекции, массивы), набор данных DataSet, документ XML. Но вне зависимости от типа источника LINQ позволяет применить ко всем один и тот же подход для выборки данных.

string[] people = { "Tom", "Bob", "Sam", "Tim", "Tomas", "Bill" };

// создаем новый список для результатов

var selectedPeople = from p in people // передаем каждый элемент из people в переменную p

                    where p.ToUpper().StartsWith("T") //фильтрация по критерию

                    orderby p  // упорядочиваем по возрастанию

                    select p; // выбираем объект в создаваемую коллекцию

foreach (string person in selectedPeople)

    Console.WriteLine(person);

**Ограничения**

**https://metanit.com/sharp/tutorial/3.38.php**

class Messenger<T> where T : Message

{

    public void SendMessage(T message)

    {

        Console.WriteLine($"Отправляется сообщение: {message.Text}");

    }

}

class Message

{

    public string Text { get; } // текст сообщения

    public Message(string text)

    {

        Text = text;

    }

}

class Messenger<T> where T : struct

{}

class Messenger<T> where T : class

{}

class Messenger<T> where T : new()

{}

class Smartphone<T> where T: Messenger, new()

{

}

**Исключения**

**https://metanit.com/sharp/tutorial/2.14.php**

try

{

}

catch

{

}

finally

{

}

**Коллекции**

**<https://metanit.com/sharp/tutorial/4.5.php>**

|  |  |
| --- | --- |
| [Dictionary<TKey,TValue>](https://learn.microsoft.com/ru-ru/dotnet/api/system.collections.generic.dictionary-2) | Предоставляет коллекцию пар «ключ-значение», которые упорядочены по ключу. |
| [List<T>](https://learn.microsoft.com/ru-ru/dotnet/api/system.collections.generic.list-1) | Представляет список объектов, доступных по индексу. Предоставляет методы для поиска по списку, его сортировки и изменения. |
| [Queue<T>](https://learn.microsoft.com/ru-ru/dotnet/api/system.collections.generic.queue-1) | Представляет коллекцию объектов, которая обслуживается в порядке поступления (FIFO). |
| [SortedList<TKey,TValue>](https://learn.microsoft.com/ru-ru/dotnet/api/system.collections.generic.sortedlist-2) | Представляет коллекцию пар "ключ-значение", упорядоченных по ключу на основе реализации [IComparer<T>](https://learn.microsoft.com/ru-ru/dotnet/api/system.collections.generic.icomparer-1). |
| [Stack<T>](https://learn.microsoft.com/ru-ru/dotnet/api/system.collections.generic.stack-1) | Представляет коллекцию объектов, которая обслуживается в обратном порядке (LIFO). |