# Лекция 2: Описание класса

1. Поля
2. Методы
3. Конструкторы

# 1. Поля

- **Поля** – переменные описанные в классе.

Синтаксис:

**[атрибут] [тип] имя [ = начальное значение]**

- По умолчанию элементы класса считаются **private**
- Для полей доступ private более предпочтительный, поскольку определяет *внутреннее устройство класса*, которое *должно быть скрыто*.
- Все *методы* класса *имеют доступ к полям.*

- *Обращение* к полю класса выполняется через **операцию доступа** (точка).
- К **константам класса** обращаются с указанием имени класса, а не имени объекта.
- К **статическим поля** обращаются по имени класса, а не имени объекта.
- Атрибут **readonly** создает поля доступ к которым возможет только для чтения по имени объекта.

# Пример использования полей

## Demo.cs

```
class Demo
{
    public int a = 5;
    public const double b = 1.55;
    public static string s = "Hello";
    double y = 10;
    static string s2 = "Hello World!";
}
```

## Program.cs

```
class Program
{
    static void Main(string[] args)
    {
        Demo X = new Demo();
        X.a = 10;
        X.a = 10.5;
        X.b = 15;
        int F = Demo.b;
        double G=Demo.b;
        X.s = "Text";
        string bufer = X.s;
        string bufer = Demo.s;
        X.y = 100;
        string bufer2 = Demo.s2;
    }
}
```

# 2. Методы

- **Метод** – функциональный элемент класса, который реализует вычисления или другие действия, выполняемые классом или экземпляром.

Синтаксис:

**[атрибут] [тип] тип имя ([тип имя]) {…}**

- Метод представляет собой законченный фрагмент кода, к которому можно обратиться по имени.
- Метод описывается один раз, а вызываться может столько раз, сколько необходимо.

- Чаще методы имеют спецификатор доступа **public** (именно с методами чаще работает пользователь, а не с полями).
- Методы объявленные как **static** вызываются по имени класса, а не по имени объекта.
- **Параметры** используются для обмена информацией с методом.
- **Сигнатура** – имя метода вкупе с количеством, типами параметров (то чем один метод отличается от другого).

# Пример использования методов

**Demo.cs**

```
class Demo
{
    private int x=50;

    public int GetX()
    {
        return x;
    }

    public void SetX(int newValue)
    {
        x = newValue;
    }

}
```

**Program.cs**

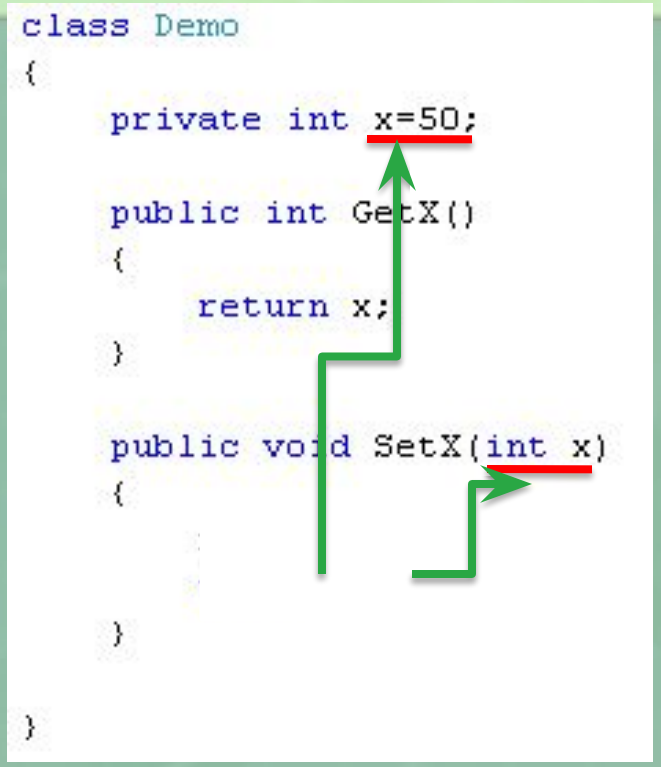
```
class Program
{
    static void Main(string[] args)
    {
        Demo X = new Demo();



    }
}
```

# Ключевое слово *this*

- Каждый объект содержит свой экземпляр полей класса. Доступ к полям класса из класса можно получить использую слово this.



# 3. Конструкторы

- **Конструктор** – предназначен для инициализации объекта.

Синтаксис:

**public имя_класса ([тип имя]) {…}**

- Вызывается автоматически при создании объекта с помощью операции **new**.
- **Имя** конструктора должно *совпадать с именем класса*.
- Конструктор *не возвращает значение*.

- Класс можете иметь несколько конструкторов *с различными параметрами* для разных видов инициализации.
- Все конструкторы должны иметь разные сигнатуры.
- Если программист не указал ни одного конструктора, то поля заполняются значениями по умолчанию или *нулями*.
- Конструктор, который вызывается без параметров – **конструктор по умолчанию**.

# Пример использования конструктора

```
class Chelovek
{

    private string name;
    private int age;
    private double ves;

    public Chelovek()
    {
        name = "Noname";
        age = 0;
        this.ves = 0;
    }

    public Chelovek(string Imya)
    {
        name = Imya;
    }

    public Chelovek(string Imya, int Age)
    {
        name = Imya;
        age = Age;
    }

}
```

```
class Program
{
    static void Main(string[] args)
    {
        Chelovek X = new Chelovek();
        Chelovek Y = new Chelovek("Степан");
        Chelovek Z = new Chelovek("Степан", 50);

    }
}
```

# 4. Перечисления

- **Перечисления** – объединяют под одним именем несколько связанных между собой и именованных констант.

Синтаксис:

**[атрибут] enum имя_перечисления [: базовый тип] {…}**

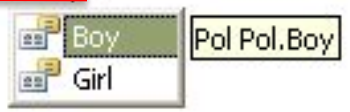
- Для каждой константы присваивается символическое имя.
- Константам по умолчанию присваиваются последовательные значения 0, 1, 2… (но можно и указывать собственные).

```
public enum Pol
{
    Boy, Girl
}
class Chelovek
{
    private Pol PolCheloveka;
    private enum Raduga {Красный, Оранжевый, Желтый, Фиолетовый =6 };

    public Chelovek(Pol kto)
    {
        PolCheloveka = kto;
    }

}
```

```
class Program
{
    static void Main(string[] args)
    {
        Chelovek X = new Chelovek(Pol.);
    }
}
```



Штрих код

QR-код