



ОНЛАЙН-ОБРАЗОВАНИЕ


Онлайн-образование

Не забыть включить запись!





Меня хорошо видно && слышно?

Ставьте  , если все хорошо
Напишите в чат, если есть проблемы

Классы как основа C#



Алена Шилина

Преподавательница курса
Middle разработчица .Net (C#)

Правила вебинара



Активно участвуем, включаем камеры



Задаем вопросы голосом или в чат



Off-topic обсуждаем в канале группы



На вопросы из чата могу ответить не сразу

Маршрут

Что такое класс в языке c#




Поля, конструктор, методы, свойства



partial, static, удобства
работы



Практика

The background of the slide is an aerial photograph of a city with many skyscrapers, overlaid with a semi-transparent blue layer. A network of white lines connects various points across the blue area, creating a digital or interconnected pattern.

Есть ли у кого-нибудь опыт
работы с классами?

The background of the slide is a blue-tinted aerial photograph of a dense city skyline, likely New York City. Overlaid on this is a semi-transparent network pattern consisting of white dots connected by thin white lines, creating a web-like effect across the center of the image.

Что такое ООП

Определение

Объектно-ориентированное программирование (ООП) – методология программирования, основанная на представлении программы в виде совокупности взаимодействующих объектов, каждый из которых является экземпляром определенного класса, а классы образуют **иерархию наследования** (Гради Буч “Объектно-ориентированный анализ и проектирование”)

Класс в объектно-ориентированном программировании – модель для создания объектов определённого типа, описывающая их структуру (набор полей и их начальное состояние) и определяющая алгоритмы (функции или методы) для работы с этими объектами.

- Программа – набор объектов
- Объекты – экземпляры класса
- Класс – комплексный тип данных, состоящий из полей и методов для работы с ними

The background of the slide is a high-angle, aerial photograph of a dense urban skyline, likely New York City, featuring numerous skyscrapers. The image is tinted with a blue and green color palette. A semi-transparent network of white lines and dots is overlaid on the image, particularly concentrated in the center where the text is located. The text "Пишем свой класс" is written in a large, white, sans-serif font across the middle of the slide.

Пишем свой класс

Пишем свой класс. Синтаксис

```
[тип доступа (public, internal, private)] class НазваниеКласса  
{  
    //код  
}
```

```
public class Car {}
```


Пишем свой класс. Создаем объекты класса

```
public class Car {}
```

```
//  
V
```

```
// Можно создавать объект класса Car - nissan
```

```
var nissan = new Car();
```

```
// lexus – ссылка на объект nissan
```

```
var lexus = nissan;
```


Пишем свой класс. Конструктор по умолчанию

```
public class Car {}
```

```
//  
V
```

```
public class Car  
{  
    public Car()  
    {  
    }  
}
```

Конструктор по умолчанию создается, если нет других конструкторов и заполняет поля класса дефолтными значениями

Пишем свой класс. Поля

```
public class Car
{
    // Трансмиссия
    private int transmission;

    // Марка машины
    public string Name;
}
```


Пишем свой класс. Создаем объекты класса

```
public class Car
{
    private int transmission;
    public string Name;
}
```

```
||
V
```

```
var nissan = new Car();
```

```
transmission = 0
Name = null
```


Пишем свой класс. Конструктор

```
public class НазваниеКласса
{
    [тип доступа] НазваниеКласса([Аргументы])
    {
        //какой-то код
    }
}
```

```
public class Car
{
    public Car(string name)
    {
        Name = name;
    }
}
```


Пишем свой класс. Создаем объекты класса

```
public class Car
{
    public string Name;
    public Car(string name)
    {
        Name = name;
    }
}
```

```
||
V
```

```
var nissan = new Car("Nissan");
```

Если есть конструктор с параметрами, то конструктор по умолчанию не создается!

Пишем свой класс. Создаем объекты класса

```
public class Car
{
    private int transmission;

    public string Name;

    public Car(string name)
    {
        Name = name;
    }
}
```

```
var nissan = new Car("Nissan");
nissan.Name = "Lexus";
nissan.transmission = 5;
```

Приватные поля доступны
только из класса, в котором
находятся!

Пишем свой класс. Методы

```
public class MyClass
{
    [тип доступа] тип_данных_ответа НазваниеМетода([параметры])
    {
        // код
    }
}
```

=>

Тип ответа void -
метод ничего не
возвращает

```
public class Car
{
    private int transmission;
    public void ChangeTransmission(int count)
    {
        transmission += count;
    }
}
```

Пишем свой класс. Методы

```
public class Car
{
    private int transmission;
    public int ChangeTransmission(int count)
    {
        transmission += count;
        return transmission;
    }
}
```

return возвращает из
метода значение

```
...
var nissan = new Car("Nissan");
Console.WriteLine($"Transmission = {nissan.ChangeTransmission(1)}");
```


Пишем свой класс. Ключевое слово this

```
public class Car
{
    private int transmission;
    public int ChangeTransmission(int count)
    {
        this.transmission += count;
        return transmission;
    }
}
```

Часто можно увидеть в
конструкторах

```
public class Car
{
    private string name;
    public Car(string name)
    {
        this.name = name;
    }
}
```

В C# - все в классах

```
internal class Program
{
    static void Main(string[] args)
    {
        Console.WriteLine("Hello World!");
    }
}
```

<https://learn.microsoft.com/en-us/dotnet/csharp/fundamentals/program-structure/top-level-statements>

<https://learn.microsoft.com/en-us/dotnet/core/tutorials/top-level-templates>

Пишем свой класс. Свойства

Свойство – особое поле, позволяющее настраивать присвоение и получение данных для него

- По принципу работы – похожи на методы
- По синтаксису – на поля
- Для получения данных – ключевое слово **get**
- Для присваивания – ключевое слово **set**

Всегда лучше **public** свойство, чем **public** поле

Пишем свой класс. Свойства

```
[тип доступа] тип_свойства НазваниеСвойства
{
    get
    {
        // Какой-то код
        return что-то;
    }
    set
    {
        // Какой-то код
        что-то = value; // value – значение извне
    }
}
```


Пишем свой класс. Свойства

```
private double _velocity = 0;
public double Velocity
{
    get { return _velocity; }
    set
    {
        if (value < 0)
            throw new InvalidOperationException("Скорость должна быть > 0");
        _velocity = value;
    }
}
```

```
nissan.Velocity = 40; //срабатывает set
Console.WriteLine(nissan.Velocity); //срабатывает get
```

Пишем свой класс. Автосвойства

```
private double _velocity;  
public double Velocity  
{  
    get { return _velocity; }  
    set { _velocity = value; }  
}
```

=>

```
public double Velocity { get; set; }
```


The background of the slide is a blue-tinted aerial photograph of a dense city skyline, likely New York City. Overlaid on this image is a semi-transparent network pattern consisting of numerous small dots connected by thin lines, creating a web-like structure. The word "Вопросы?" is centered in the middle of the slide in a large, white, sans-serif font.

Вопросы?

Какая ошибка допущена в программе?

```
1  using System;
2
3  public class Person
4  {
5      public string Name { get; set; }
6      public Person(string name)
7      {
8          Name = name;
9      }
10 }
11
12 public class Program
13 {
14     public static void Main(string[] args)
15     {
16         var person = new Person();
17         person.Name = "Anna";
18     }
19 }
```


Какая ошибка допущена в программе?

```
1  using System;
2
3  public class Person
4  {
5      private string _name;
6      public string Name
7      {
8          get { return Name; }
9          set { Name = value; }
10     }
11     public Person(string name)
12     {
13         Name = name;
14     }
15 }
```

```
17 public class Program
18 {
19     public static void Main(string[] args)
20     {
21         var person = new Person("Anna");
22         person.Name = "Elena";
23     }
24 }
```


The background of the slide is a blue-tinted aerial photograph of a dense city skyline, likely New York City. Overlaid on this image is a semi-transparent network pattern consisting of numerous small dots connected by thin lines, creating a web-like structure. The text is centered within a horizontal band that transitions from a lighter blue on the left to a darker blue on the right.

partial, static, удобства работы

partial

1. Класс очень большой, часть методов/свойств/полей можно семантически объединить вместе
2. С одним классом работают несколько программистов
3. Часть кода генерируется автоматически – обычно создается в отдельном файле

```
// В одном файле Car.cs
public partial class Car
{
    public Car(string name)
    {
        Name = name;
    }
    // код
}
```

```
// В другом файле CarActions.cs
public partial class Car
{
    public void StartEngine()
    {
    }

    public void Drive()
    {
    }
}
```

static. Когда используется

1. Есть методы/функции, которые могут работать без объекта, содержащих их
2. Есть набор констант, доступных в разных частях приложения
3. Есть объект, который будет всегда в единственном экземпляре
4. Надо «расширить» класс новыми методами

static. Статический класс

Статический класс – класс, из которого нельзя создавать объекты

```
static class MathStatic
{
    public static int Pow(int a, int p)
    {
        return 0
    }
}
```

// Все ошибка

```
var m = new MathStatic();
Console.WriteLine(m.Pow(2, 4));
```

// Но можно так

```
Console.WriteLine(MathStatic.Pow(2, 4));
```

Статические классы могут содержать только статические члены класса

static. Статический конструктор

Статические конструкторы не должны иметь модификатор доступа и не принимают параметров, их нельзя вызвать вручную

```
static class MathStatic
{
    //автоматически вызывается при первом обращении к классу
    static MathStatic()
    {
        Console.WriteLine("Я статический конструктор");
    }
}
```

Статический конструктор может быть и у нестатического класса

static. Статические методы, поля, свойства

```
class MathStatic
{
    public int MyValue = 5;

    public static double Pi = 3.141592;

    public static string MyProp { get; set; }

    public static int Pow(int a, int p)
    {
        MyValue = 90; //так нельзя
        Pi = 90; //так можно
    }
}
```

Статические методы могут обращаться
только к статическим членам класса

static. Статические методы, поля, свойства

```
class MathStatic
{
    public int MyValue = 5;

    public static double Pi = 3.141592;

    public static string MyProp { get; set; }

    public static int Pow(int a, int p)
    {
        //код
    }
}
```

```
Console.WriteLine(MathStatic.Pi);
MathStatic.Pow(2,3);
```

К статическим членам классам обращение идет по имени класса, а не по имени экземпляра как обычно

static. Методы расширения

```
public static class CarExtensions
{
    public static void Drive(this Car car, int speed)
    {
        car.Velocity = speed;
        Console.WriteLine("VRUM!");
    }
}
```

```
var nissan = new Car("Nissan");
nissan.Drive(10);
```

```
// Или так
CarExtensions.Drive(nissan, 10);
```


The background of the slide is a blue-tinted aerial photograph of a dense city skyline, likely New York City. Overlaid on this is a semi-transparent network pattern consisting of numerous small dots connected by thin lines, creating a web-like structure across the center of the image.

Вопросы?

Какая ошибка допущена в программе?

```
1  using System;
2
3  public class Person
4  {
5      private string _name;
6      public string Name
7      {
8          get { return _name; }
9          set { _name = value; }
10     }
11     static Person(string name)
12     {
13         Name = name;
14     }
15 }
```

```
17 public class Program
18 {
19     public static void Main(string[] args)
20     {
21         var person = new Person("Anna");
22         person.Name = "Elena";
23     }
24 }
```

Какая ошибка допущена в программе?

```
1  using System;
2
3  public class Person
4  {
5      private string _name;
6      public string Name
7      {
8          get { return _name; }
9          set { _name = value; }
10     }
11     public static string ChangeName(string name)
12     {
13         this.Name = name;
14         return this.Name;
15     }
16 }
```

```
18 public class Program
19 {
20     public static void Main(string[] args)
21     {
22         var person = new Person();
23         Person.ChangeName("Elena");
24     }
25 }
```


Удобства работы. Классы внутри классов

```
public class Car
{
    // Класс внутри, но доступен снаружи
    public class Engine { }

    // доступен только внутри Car
    class Tier { }
}
```

```
// Так - можно
var v8 = new Car.Engine();
```

```
// Так - нельзя
var michelin = new Car.Tier();
```

Удобства работы. Разные модификаторы доступа

Расположение вызывающего объекта	public	protected internal	protected	internal	private protected	private	file
В файле	✓	✓	✓	✓	✓	✓	✓
В классе	✓	✓	✓	✓	✓	✓	✗
Производный класс (та же сборка)	✓	✓	✓	✓	✓	✗	✗
Непроизводный класс (та же сборка)	✓	✓	✗	✓	✗	✗	✗
Производный класс (другая сборка)	✓	✓	✓	✗	✗	✗	✗
Непроизводный класс (другая сборка)	✓	✗	✗	✗	✗	✗	✗

Удобства работы. Разные модификаторы доступа

```
private double _velocity = 0;  
public double Velocity  
{  
    public get { return _velocity; }
```

```
private set
```

```
{  
    if (value < 0)  
        throw new InvalidOperationException("Скорость должна быть > 0");  
    _velocity = value;  
}  
}
```

Устанавливать значение свойству можно
только внутри класса

Удобства работы. Только get или set

```
private double _velocity = 0;
public double Velocity
{
    set
    {
        _velocity = value;
    }
}
```

```
private double _velocity = 0;
public double Velocity
{
    get
    {
        return _velocity;
    }
}
```


Удобства работы. Упрощаем get/set

```
private double _velocity = 0;
public double Velocity
{
    get
    {
        return _velocity;
    }
    set
    {
        _velocity = value;
    }
}
```

```
public double Velocity
{
    get => _velocity;
    set => _velocity =value;
}
```

Удобства работы. Упрощаем get/set

```
public double Exponent  
{  
    get  
    {  
        return 2.17;  
    }  
}
```

```
public double Exponent => 2.17;
```


The background of the slide is a blue-tinted aerial photograph of a dense city skyline, likely New York City. Overlaid on this image is a semi-transparent network pattern consisting of numerous small dots connected by thin lines, creating a web-like structure. The word "Практика" is centered in the middle of the slide in a large, white, sans-serif font.

Практика

Задача

Некое производство выпускает вазы, все одинакового объема.

В вазу можно налить воды, вылить воду и поставить цветы.
Цветы можно поставить только тогда, когда ваза наполнена.

При покупке вазы можно выбрать ее цвет.

The background of the slide is an aerial photograph of a dense city skyline, likely New York City, with numerous skyscrapers. The image is overlaid with a semi-transparent blue layer. A network of thin, light blue lines connects various points across the blue area, creating a digital or technological aesthetic. The text 'Домашняя работа' is centered in a large, white, sans-serif font.

Домашняя работа

The background of the slide is an aerial photograph of a dense city skyline, likely New York City, with numerous skyscrapers. The image is overlaid with a semi-transparent blue layer. A network of thin, light blue lines connects various points across the blue area, creating a digital or technological aesthetic. The text is centered in white, bold, sans-serif font.

Заполните, пожалуйста,
опрос о занятии по ссылке в чате

The background of the slide is a blue-tinted aerial photograph of a dense city skyline, likely New York City. Overlaid on this is a semi-transparent network pattern consisting of numerous small dots connected by thin lines, creating a web-like structure across the center of the image.

Вопросы?

Спасибо за занятие!



Алена Шилина

Преподавательница курса
Middle разработчица .Net (C#)