Основные принципы объектно-ориентированного программирования (ООП)

Основные конструкции языка C#

Поля и методы класса

```
class Employee
{
    public string name; //поле
    public int ID;      //поле
    ссылка: 1
    public void Print() //метод
    {
        Console.WriteLine($"Имя сотрудника: {name}\nID: {ID}");
    }
}
```

Конструктор класса

Общий синтаксис:
new конструктор_класса(параметры_конструктора);

```
class Employee
{
    public string name; //поле
    public int ID;      //поле

    ссылка: 1
    public Employee()   //конструктор класса
    {
        name ="undefined";
        ID = 000;
        Console.WriteLine("объект класса создан");
    }

    ссылка: 1
    public void Print()
    {
        Console.WriteLine($"Имя сотрудника: {name}\nID: {ID}");
    }
}
```

Инициализирует поля объекта и выводит сообщение

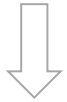
```
Employee emp = new Employee();
emp.Print();
```

Выбрать Консоль отладки Microsoft Visual Studio

```
объект класса создан
Имя сотрудника: undefined
ID: 0
```

Разные типы конструкторов внутри одного класса

```
class Employee
{
    public string name;      //поле
    public int ID;           //поле

    ссылка: 1
    public Employee()        //конструктор без параметров...
    Ссылок: 0
    public Employee(int n)              {ID = n; }              //конструктор с 1 параметром
    Ссылок: 0
    public Employee(string s)           { name = s; ID =0; }    //конструктор с 1 параметром
    Ссылок: 0
    public Employee(int n, string s)    { ID=n; name = s; }     //конструктор с 2 параметрами
    ссылка: 1
    public void Print()      //метод...

}
```



```
Employee employee1 = new Employee(1024);
employee1.Print();
Employee employee2 = new Employee(1025,"Василий");
employee2.Print();
Employee employee3 = new Employee("Светлана");
employee3.Print();
```

```
Консоль отладки Microsoft Visual Studio
Имя сотрудника:
ID: 1024
Имя сотрудника: Василий
ID: 1025
Имя сотрудника: Светлана
ID: 0
```

Ключевое слово **this**

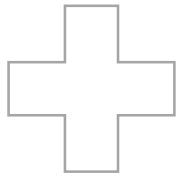
- Это ссылка на текущий экземпляр класса

```
class Employee
{
    public string name;      //поле
    public int ID;           //поле
    Ссылок: 3
    public Employee(int ID, string name) { this.ID = ID; this.name = name; }
}
```

Инициализаторы объектов

представляют передачу в фигурных скобках значений доступным полям и свойствам объекта: `Employee e = new Employee { ID = 1011, name = "Aleksey" };`

```
class Employee
{
    public int ID;
    public string name;
    public Company company;

    Ссылок: 0
    public Employee()
    {
        ID = 0; name = "Undefined";
        company = new Company();
    }

}
```



```
public class Company
{
    public string title = "unknow";
}
```

```
Employee employee = new Employee {name = "Aleksey", company = {title = "OKEI" }
```

Модификаторы доступа

Сборка - один или несколько файлов, содержащих все необходимые сведения о развертывании программы и ее версии

Модификаторы доступа позволяют задать область видимости (переменных, объектов, полей и т.д.)

модификаторы	Текущий класс	Производный класс из текущей сборки	Производный класс из другой сборки	Непроизводный класс из текущей сборки	Непроизводный класс из бругой сборки
Private					
Private protected					
Protected					
Internal					
Protected internal					
public					

Модификаторы по умолчанию

```
class Person //internal
{
    string name; //private
    Ссылок: 0
    public Person(string name)
    {
        this.name = name;
    }
    Ссылок: 0
    void Print() //private
    {
        Console.WriteLine($"Name: {name}");
    }
}
```

Свойства

Общий синтаксис:

[модификаторы] тип_свойства название_свойства
{
get { действия, выполняемые при получении значения свойства}
set { действия, выполняемые при установке значения свойства}
}

аксессоры

```
class Person
{
    private string name = "Undefined";
    Ссылок: 0
    public string Name
    {
        get { return name; }    // возвращаем значение свойства

        set { name = value; }   // устанавливаем новое значение свойства
    }
}
```

Свойства только для чтения и записи

Допускается использовать только один аксессор в свойстве, тогда оно будет доступно только на чтение или на запись

```
class Person
{
    string name = "Tom";
    int age = 1;
    Ссылок: 0
    public int Age // свойство только для записи
    {
        set { age = value; }
    }
    Ссылок: 0
    public string Name // свойство только для чтения
    {
        get { return name; }
    }
}
```

автосвойства

- Используются для сокращенной записи
- Поля автоматически генерируются при компиляции

```
class Person
{
    ссылка: 1
    public string Name { get; set; }
    ссылка: 1
    public int Age { get; set; }
    Ссылок: 0
    public Person(string name, int age)
    {
        Name = name;
        Age = age;
    }
}
```

Сокращенная запись свойств

Сокращение аксессоров

```
class Person
{
    string name;
    Ссылок: 0
    public string Name
    {
        get => name;
        set => name = value;
    }
}
```

Сокращение всего свойства

```
class Person
{
    string name;
    Ссылок: 0
    public string Name => name;
}
```

Реализация наследования

Конструкторы базового класса на наследуются!

Базовый класс

```
class Person
{
    private string _name = "";
    Ссылок: 2
    public string Name
    {
        get { return _name; }
        set { _name = value; }
    }
    Ссылок: 0
    public void Print()
    {
        Console.WriteLine(Name);
    }
}
```

Унаследованный класс

```
class Student : Person
{

}
```

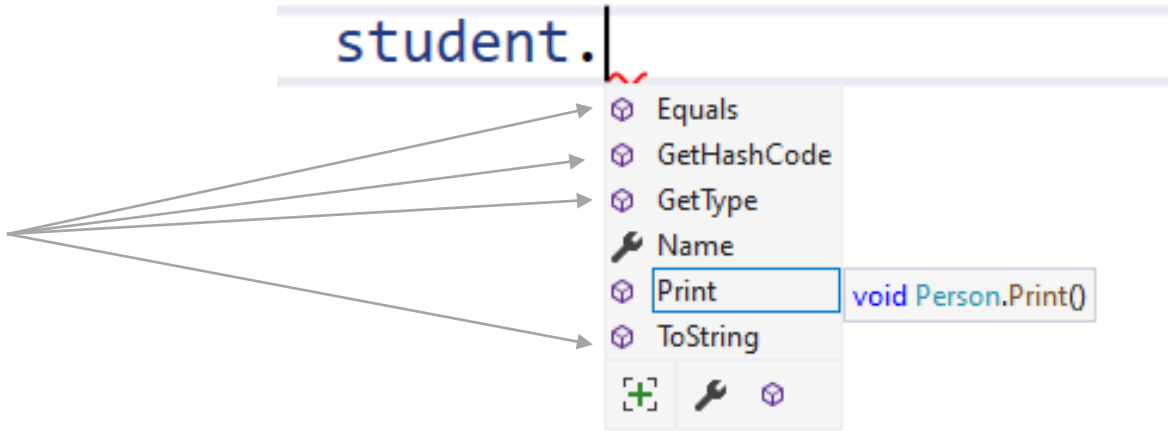
```
static void Main(string[] args)
{
    Person person = new Person { Name = "Aleksey" };
    person.Print();

    Student student = new Student { Name = "Ivan" };
    student.Print();
}
```

Консоль отладки Microsoft Visual Studio

```
Aleksey
Ivan
```

По умолчанию все классы наследуются от базового класса Object,

Методы, унаследованные по умолчанию от класса Object



Ограничения наследования

1. Нет множественного наследования
2. Не допускается у дочернего класса применять более открытый модификатор доступа
3. От класса с модификатором sealed нельзя наследовать
4. Нельзя наследовать от статического класса

Доступ к содержимому родительского класса из дочернего

Базовый класс

```
class Person
{
    private string _name = "";
    Ссылок: 2
    public string Name
    {
        get { return _name; }
        set { _name = value; }
    }
    Ссылок: 0
    public void Print()
    {
        Console.WriteLine(Name);
    }
}
```

```
class Student : Person
{
    Ссылок: 0
    public void PrintName()
    {
        Console.WriteLine(_name);
    }
}
```

class System.String
Represents text as a sequence of UTF-16 code units.

'Person._name' недоступен из-за его уровня защиты.

Показать возможные решения (Alt+ВВОДилиCtrl+ю)

```
class Student : Person
{
    Ссылок: 0
    public void PrintName()
    {
        Console.WriteLine(Name);
    }
}
```

base

Базовый класс

```
class Person
{
    Ссылок: 4
    public string Name { get; set; }
    Ссылок: 2
    public Person(string name)
    {
        Name = name;
    }
    Ссылок: 2
    public void Print()
    {
        Console.WriteLine(Name);
    }
}
```

Унаследованный класс

```
class Student : Person
{
    ссылка: 1
    public string Group { get; set; }
    ссылка: 1
    public Student(string name, string group)
        : base(name)
    {
        Group = group;
    }
}
```

```
Person person = new Person("Aleksey");
person.Print();

Student student = new Student("Ivan","1pk2");
student.Print();
```