# Наследование

class Person **{** string Name **{** **get;** **set;** **}** **}**

class Employee **:** Person **{** **...** **}**

Person person **=** **new** Person **{** Name **=** "Tom" **};**

Person employee **=** **new** Employee **{** Name **=** "Sam" **};**

*Ограничения****:***

**-** Класс может наследоваться только от одного класса**.**

**-** Тип доступа к производному классу должен быть таким же**,** как и у базового класса**,** или более строгим**.** Если базовый и производный класс находятся в разных сборках **(**проектах**),** то производый класс может наследовать только от класса **public.**

**-** **sealed** класс не допускает создание наследников**:** **sealed** class Admin **{}**

**-** Нельзя унаследовать класс от статического класса**.**

## Ключевое слово base

Если в базовом классе не определен конструктор по умолчанию без параметров**,** а только конструкторы с параметрами**,** то в производном классе мы обязательно должны вызвать один из этих конструкторов через ключевое слово **base.**

При вызове конструктора класса сначала отрабатываются конструкторы базовых классов и только затем конструкторы производных**.**

class Person

**{**

**public** string Name **{** **get;** **set;}**

**public** Person**(**string name**)**

**{**

Name **=** name**;**

**}**

**}**

class Employee **:** Person

**{**

**public** string Company **{** **get;** **set;** **}**

**public** Employee**(**string name**,** string company**)** **:** **base(**name**)**

**{**

Company **=** company**;**

**}**

**}**

# Преобразование типов

## Восоходящее преобразование. Upcasting

Employee employee **=** **new** Employee**(**"Tom"**,** "Microsoft"**);**

Person person **=** employee**;** // преобразование от Employee к Person

employee и person будут указывать на один и тот же объект в памяти**,** но переменной person будет доступна только та часть**,** которая представляет функционал типа Person**.**

## Нисходящее преобразование. Downcasting

Person person **=** **new** Person**();**

Employee employee2 **=** **(**Employee**)**person**;** // преобразование от Person к Employee

## Способы преобразований

Person person **=** **new** Person**();**

Employee**?** employee **=** person **as** Employee**;** // В случае неудачного преобразования выражение будет содержать значение null

**if** **(**person **is** Employee employee**)**

**{**

Console**.**WriteLine**(**employee**.**Company**);**

**}**

Выражение **if** **(**person **is** Employee employee**)** проверяет**,** является ли переменная person объектом типа Employee**.** Если person является объектом Employee**,** то автоматически преобразует значение переменной person в тип Employee и преобразованное значение сохраняет в переменную employee**.**

**if** **(**person **is** Employee**)** // без преобразования типа Person в Employee

**{**

Console**.**WriteLine**(**"Представляет тип Employee"**);**

**}**

# Виртуальные методы и свойства

Методы в родительском классе**,** помеченные модификатором **virtual,** доступны для переопределения**.**

В классе**-**наследнике мы можем переопределить его**,** пометив модификатором **override.**

class Person

**{**

**virtual** void Method**(){}**

**}**

class Employee **:** Person

**{**

**virtual** void Method**(){}**

**}**

Ограничения**:**

**-** Виртуальный и переопределенный методы должны иметь один и тот же модификатор доступа**.**

**-** Нельзя переопределить или объявить виртуальным статический метод**.**

## Ключевое слово base

class Employee **:** Person

**{**

Employee**(** **...** **)** **:base(** **...** **)** // из базового класса можно вызвать и конструктор

**{**

**...**

**}**

**override** void Method**()**

**{**

**base.**Print**();** // и метод с помощью ключевого слова base

**...**

**}**

**}**

## Переопределение свойств

Можно переопределять свойства**:**

class Person

**{**

**private** int age**;**

**public** **virtual** int Age **{** **get;** **set;** **}**

**}**

class Employee **:** Person

**{**

**public** **override** Age

**{**

**get** **=>** **base.**Age**;**

**set** **{** **if** **(value** **>** 18**)** **base.**Age **=** **value;** **}**

**}**

**}**

## Запрет переопределения методов

**override** **sealed** запрещает переопределение методов и свойств**.** Оно работает только в уже переопределенных методах в классах**-**наследниках**,** т**.**к**.** в родительском классе достаточно не ставить модификатор **virtual:**

class Employee **:** Person

**{**

**override** **sealed** void Method**()** **{** **...** **}**

**}**