**Абстрактный класс**

Абстрактный класс-класс, объявляенный ключевым словом abstract

abstract class[имя класса]

{

//тело

}

Имеет ключевые особенности

1. Нельзя создать объекты(экземпляры) класса
2. Абстрактный класс может содержать как абстрактный методы/свойства так и обычные
3. В классе наследнике должны быть реализованы все абстрактные методы и свойства, объявленные в базовом классе

Зачем нужны абстрактные классы?

В самом по себе абстрактном классе, от которого никто не наследуется, смысла нет, так как нельзя создавать его экземпляры

В абстрактном классе обычно реализуется некоторая общая часть нескольких сущностей или другими словами абстрактная сущность, которая, как объект, не может существовать, и эта часть необходима в классе наследнике.

Абстрактный метод- метод, который не имеет своей реализации в базовом классе, и он должен быть реализован в классе наследнике.

Абстрактный метод может быть объявлен только в абстрактном классе.

Объявляение абстрактного метода происходит при помощи ключевого слова abstract , и при этом фигурные скобки опускаются , точка с запятой ставится после заголовка метода.

**[модификатор доступа] abstract [тип] [имя метода] ([аргументы])**

Реализация метода в классе наследнике происходит также как и переопределение метода –при помощи ключевого слова **override**

**[модификатор доступа] override [тип] [имя метода] ([аргументы])**

**{**

**//реализация метода**

**}**

Абстрактные свойства объявляются аналогично абстрактным методам

При наследовании нередко возникает необходимость изменить в классе-наследнике функционал метода, который был унаследован от базового класса. В этом случае класс-наследник может **переопределять методы и свойства базового класса**.

Те методы и свойства , которые мы хотим сделать доступными для переопределения, в базовом классе помечаются модификатором **virtual** . Такие методы и свойства называются **виртуальными**.

А чтобы переопределить метод в классе наследнике этот метод помечается модификатором **override**. Переопределенный метод в классе-наследнике должен иметь тот же набор параметров, что и виртуальный метод в базовом классе

**Интерфейс**

**Интерфейс-**поименованный набор сигнатур методов. В интерфейсе нельзя определить конструкторы и поля, а также статические поля и константы.

Интерфейс же в отличии от класса содержит в себе только определение сигнатур метода, без их непосредственной реализации

Для создания интерфейса используется ключевое слово **interface**

Для именования интерфейсов принято соглашение об использовании заглавной буквы I в начале имени.

Если какой либо класс решит реализовать интерфейс-это будет означать, что этот класс обязан реализовать все методы, определенные в интерфейсе.

**Interface ISomeInterface**

**{**

**//тело интерфейса**

**}**

Чтобы указать, что класс реализует интерфейс, необходимо, так же, как и при наследовании, после имени класса и двоеточия указать имя интерфейса:

**Class SomeClass : ISomeInterface //реализация интерфейса ISomeInterface**

**{**

**//тело класса**

**}**

Множественное наследованиеесть в языке С++, а в С# от него отказались и внесли интерфейсы. В С# класс может реализовать сразу несколько интерфейсов.

**Нельзя создавать объекты интерфейса напрямую с помощью конструктора, как например, в классах:**

**Interface IMovable**

**{**

**Void Move();**

**}**

**Интерфейс предназначен для реализации в классах.**

**//применение интерфейса в классе**

**Class Person: IMovable**

**{**

**Public void Move()**

**{**

**Console.WriteLine(“Человек идет”);**

**}**

**}**