**Абстрактные классы**

Некоторые сущности, которые мы хотим выразить с помощью языка программирования, могут не иметь конкретного воплощения. Например, в реальности не существует геометрической фигуры как таковой. Есть круг, прямоугольник, квадрат, но просто фигуры нет. Однако же и круг, и прямоугольник имеют что-то общее и являются фигурами. И для описания подобных сущностей, которые не имеют конкретного воплощения, предназначены абстрактные классы.

Абстрактный класс похож на обычный класс. Он также может иметь переменные, методы, конструкторы, свойства. Единственное, что при определении абстрактных классов используется ключевое слово abstract. Например, определим абстрактный класс, который представляет некое транспортное средство:

abstract class Transport

{

public void Move()

{

Console.WriteLine("Транспортное средство движется");

}

}

Транспортное средство представляет некоторую абстракцию, которая не имеет конкретного воплощения. То есть есть легковые и грузовые машины, самолеты, морские судна, кто-то на космическом корабле любит покататься, но как такового транспортного средства нет. Тем не менее все транспортные средства имеют нечто общее - они могут перемещаться. И для этого в классе определен метод Move, который эмулирует перемещение.

Но главное отличие абстрактных классов от обычных состоит в том, что мы НЕ можем использовать конструктор абстрактного класса для создания экземпляра класса.

**Абстрактные члены классов**

Кроме обычных свойств и методов абстрактный класс может иметь абстрактные члены классов, которые определяются с помощью ключевого слова abstract и не имеют никакого функционала. В частности, абстрактными могут быть:

Методы

Свойства

Индексаторы

События

Абстрактные члены классов не должны иметь модификатор private. При этом производный класс обязан переопределить и реализовать все абстрактные методы и свойства, которые имеются в базовом абстрактном классе. При переопределении в производном классе такой метод или свойство также объявляются с модификатором override (как и при обычном переопределении виртуальных методов и свойств). Также следует учесть, что если класс имеет хотя бы одный абстрактный метод (или абстрактные свойство, индексатор, событие), то этот класс должен быть определен как абстрактный.

Абстрактные члены также, как и виртуальные, являются частью полиморфного интерфейса. Но если в случае с виртуальными методами мы говорим, что класс-наследник наследует реализацию, то в случае с абстрактными методами наследуется интерфейс, представленный этими абстрактными методами.