Интерфейсы

Интерфейс представляет ссылочный тип, который может определять некоторый функционал - набор методов и свойств без реализации. Затем этот функционал реализуют классы и структуры, которые применяют данные интерфейсы.

Определение интерфейса

Для определения интерфейса используется ключевое слово **interface**. Как правило, названия интерфейсов в C# начинаются с заглавной буквы **I**, например, IComparable, IEnumerable (так называемая венгерская нотация), однако это не обязательное требование, а больше стиль программирования.

Что может определять интерфейс? В целом интерфейсы могут определять следующие сущности:

- Методы
- Свойства
- Индексаторы
- События
- Статические поля и константы (начиная с версии C# 8.0)

Однако интерфейсы не могут определять нестатические переменные. Например, простейший интерфейс, который определяет все эти компоненты:

```
interface IMovable
{
    // константа
    const int minSpeed = 0;     // минимальная скорость
    // статическая переменная
    static int maxSpeed = 60;   // максимальная скорость
    // метод
    void Move();                // движение
    // свойство
    string Name { get; set; }   // название
    delegate void MoveHandler(string message);  // определение делегата для события
    // событие
    event MoveHandler MoveEvent;    // событие движения
}
```

Интерфейсы

В данном случае определен интерфейс IMovable, который представляет некоторый движущийся объект. Данный интерфейс содержит различные компоненты, которые описывают возможности движущегося объекта. То есть интерфейс описывает некоторый функционал, который должен быть у движущегося объекта.

Методы и свойства интерфейса могут не иметь реализации, в этом они сближаются с абстрактными методами и свойствами абстрактных классов. В данном случае интерфейс определяет метод Move, который будет представлять некоторое передвижение. Он не имеет реализации, не принимает никаких параметров и ничего не возвращает.

То же самое в данном случае касается свойства Name. На первый взгляд оно похоже на автоматическое свойство. Но в реальности это определение свойства в интерфейсе, которое не имеет реализации, а не автосвойство.

Еще один момент в объявлении интерфейса: если его члены - методы и свойства не имеют модификаторов доступа, но фактически по умолчанию доступ **public**, так как цель интерфейса - определение функционала для реализации его классом. Это касается также и констант и статических переменных, которые в классах и структурах по умолчанию имееют модификатор private. В интерфейсах же они имеют по умолчанию модификатор public. И например, мы могли бы обратиться к константе minSpeed и переменной maxSpeed интерфейса IMovable:

```
static void Main(string[] args)
{
    Console.WriteLine(IMovable.maxSpeed);
    Console.WriteLine(IMovable.minSpeed);
}
```

Но также, начиная с версии C# 8.0, мы можем явно указывать модификаторы доступа у компонентов интерфейса:

```
interface IMovable
{
    public const int minSpeed = 0;      // минимальная скорость
    private static int maxSpeed = 60;   // максимальная скорость
    public void Move();
    protected internal string Name { get; set; }    // название
    public delegate void MoveHandler(string message);  // определение делегата для события
    public event MoveHandler MoveEvent;    // событие движения
}
```

Интерфейсы

Начиная с версии C# 8.0 интерфейсы поддерживают реализацию методов и свойств по умолчанию. Это значит, что мы можем определить в интерфейсах полноценные методы и свойства, которые имеют реализацию как в обычных классах и структурах. Например, определим реализацию метода Move по умолчанию:

```
interface IMovable
{
    // реализация метода по умолчанию
    void Move()
    {
        Console.WriteLine("Walking");
    }
}
```

С реализацией свойств по умолчанию в интерфейсах дело обстоит несколько сложнее, поскольку мы не можем определять в интерфейсах нестатические переменные, соответственно в свойствах интерфейса мы не можем манипулировать состоянием объекта. Тем не менее реализацию по умолчанию для свойств мы тоже можем определять:

```
interface IMovable
{
    void Move()
    {
        Console.WriteLine("Walking");
    }
    // реализация свойства по умолчанию
    // свойство только для чтения
    int MaxSpeed { get { return 0; } }
}
```

Интерфейсы

Стоит отметить, что если интерфейс имеет приватные методы и свойства (то есть с модификатором private), то они должны иметь реализацию по умолчанию. То же самое относится к любым статическим методам и свойствам (не обязательно приватным):

```
interface IMovable
{
    public const int minSpeed = 0;     // минимальная скорость
    private static int maxSpeed = 60;   // максимальная скорость
        // находим время, за которое надо пройти расстояние distance со скоростью speed
    static double GetTime(double distance, double speed) => distance / speed;
    static int MaxSpeed
    {
        get { return maxSpeed; }
        set
        {
            if (value > 0) maxSpeed = value;
        }
    }
}
class Program
{
    static void Main(string[] args)
    {
        Console.WriteLine(IMovable.MaxSpeed);
        IMovable.MaxSpeed = 65;
        Console.WriteLine(IMovable.MaxSpeed);
        double time = IMovable.GetTime(100, 10);
        Console.WriteLine(time);
    }
}
```

Интерфейсы

Модификаторы доступа интерфейсов

Как и классы, интерфейсы по умолчанию имеют уровень доступа **internal**, то есть такой интерфейс доступен только в рамках текущего проекта. Но с помощью модификатора public мы можем сделать интерфейс общедоступным:

```
public interface IMovable
{
    void Move();
}
```