Міністерство освіти і науки України
Криворізький коледж Національного авіаційного університету

Курс лекцій

з предмету

“Об’єктно-орієнтоване програмування”

Спеціальність 5.05010301

м. Кривий Ріг

Лекция №4

Тема: Язык C#

Вопросы, рассматриваемые на лекции

- **Интерфейсы**

Интерфейсы

- ***Интерфейс*** определяет набор методов, которые будут реализованы классом. Сам интерфейс не реализует методы. Таким образом, интерфейс — это логическая конструкция, которая описывает методы, не устанавливая жестко способ их реализации.
- В объектно-ориентированном программировании иногда требуется определить, *что* класс должен делать, а не *как* он будет это делать.
- Для реализации интерфейса класс должен обеспечить тела (способы реализации) методов, описанных в интерфейсе. Каждый класс может определить собственную реализацию. Таким образом, два класса могут реализовать один и тот же интерфейс различными способами, но все классы поддерживают одинаковый набор методов. Следовательно, код, "осведомленный" о наличии интерфейса, может использовать объекты любого класса, поскольку интерфейс для всех объектов одинаков. Предоставляя программистам возможность применения такого средства программирования, как интерфейс, C# позволяет в полной мере использовать аспект полиморфизма, выражаемый как "один интерфейс — много методов".
-

Интерфейсы

Интерфейсы объявляются с помощью ключевого слова interface. Вот как выглядит упрощенная форма объявления интерфейса:

```
interface имя{
тип_возврата имя_метода1 (список_параметров) ;
тип_возврата имя_метода2 (список_параметров) ;
// . . .
тип_возврата имя_методаN (список_параметров) ;
}
```

Имя интерфейса задается элементом *имя.* Методы объявляются с использованием лишь типа возвращаемого ими значения и сигнатуры. Все эти методы, по сути, — абстрактные. Как упоминалось выше, для методов в интерфейсе не предусмотрены способы реализации. Следовательно, каждый класс, который включает интерфейс, должен реализовать все его методы. В интерфейсе методы неявно являются открытыми (public-методами), при этом не разрешается явным образом указывать спецификатор доступа.

Интерфейсы

- Помимо сигнатур методов интерфейсы могут объявлять сигнатуры свойств, индексаторов и событий
- Интерфейсы не могут иметь членов данных.
- Они не могут определять конструкторы, деструкторы или операторные методы.
- Кроме того, ни один член интерфейса не может быть объявлен статическим.

Реализация интерфейсов

Один и тот же интерфейс может быть реализован одним или несколькими классами.

Чтобы реализовать интерфейс, нужно указать его имя после имени класса подобно тому, как при создании производного указывается базовый класс.

Формат записи класса, который реализует интерфейс, имеет вид:

```
class имя_класса : имя__интерфейса
{
// тело класса
}
```

Имя реализуемого интерфейса задается с помощью элемента *имя_интерфейса.*

Если класс реализует интерфейс, он должен это сделать в полном объеме, т.е. реализация интерфейса не может быть выполнена частично.

В классах, которые реализуют интерфейсы, можно определять дополнительные члены.

Реализация интерфейсов

- Классы могут реализовать несколько интерфейсов. В этом случае имена интерфейсов отделяются запятыми.
- Класс может наследовать базовый класс и реализовать один или несколько интерфейсов. В этом случае список интерфейсов должно возглавлять имя базового класса.
- Методы, которые реализуют интерфейс, должны быть объявлены открытыми. Так как методы внутри интерфейса неявно объявляются открытыми, поэтому их реализации также должны быть открытыми.
- Кроме того, сигнатура типа в реализации метода должна в точности совпадать с сигнатурой типа, заданной в определении интерфейса.

Пример

```
 using System;
public interface A
{
    void methl();
    void meth2();
}
    // Интерфейс B теперь включает методы methl() и meth2(),
// а также добавляет метод meth3().
public interface B : A
{
    void meth3();
}
```

Пример (продолжение)

```
// Этот класс должен реализовать все методы
// интерфейсов A и B.
class MyClass : B
{
    public void methl()
    {       Console.WriteLine("Реализация метода methl().");    }
    public void meth2()
    {       Console.WriteLine("Реализация метода meth2().");    }
    public void meth3()
    {       Console.WriteLine("Реализация метода meth3().");    }
}
class IFExtend
{
    public static void Main()
    {      MyClass ob = new MyClass();
        ob.methl();        ob.meth2();        ob.meth3();
    }
}
```

Явная реализация членов интерфейса

При реализации члена интерфейса можно квалифицировать его имя с использованием имени интерфейса. В этом случае говорят, что *член интерфейса реализуется явным образом,* или имеет место его *явная реализация.*

Например,

```
class MyClass : IMyIF
{
int IMyIF.myMeth(int x) { return x *x;}
}
```

Явная реализация членов интерфейса может понадобиться по двум причинам. Во-первых, можно обозначить части закрытой реализации, которые не "видны" коду, определенному вне класса. Во-вторых, класс может реализовать два интерфейса, которые объявляют методы с одинаковыми именами и типами. Полная квалификация имен позволяет избежать неопределенности ситуации.

Пример

```
// Использование явной реализации для того, чтобы избежать
// неоднозначности.
using System;
interface IMyIF_A {  int meth(int x);  }
interface IMyIF_B {  int meth(int x);  }
// В классе MyClass реализованы оба интерфейса,
class MyClass : IMyIF_A, IMyIF_B
{
    // Явным образом реализуем два метода meth().
    int IMyIF_A.meth(int x)
    {       return x + x;    }
    int IMyIF_B.meth(int x)
    {       return x * x;    }
```

Пример (продолжение)

```
// Вызываем метод meth() посредством ссылки на интерфейс,
    public int methA(int x)
    {
        IMyIF_A a_ob;
        a_ob = this;
        return a_ob.meth(x); // Имеется в виду интерфейс IMyIF_A.
    }
    public int methB(int x)
    {
        IMyIF_B b_ob;
        b_ob = this;
        return b_ob.meth(x); // Имеется в виду интерфейс IMyIF_B
    }
}
```

Пример (продолжение)

```
class FQIFNames
{
    public static void Main()
    {
        MyClass ob = new MyClass();
        Console.Write("Вызов метода IMyIF_A.meth(): ");
        Console.WriteLine(ob.methA(3));
        Console.Write("Вызов метода IMyIF_B.meth(): ");
        Console.WriteLine(ob.methB(3));
    }
}
```