



Интерфейсы

Последнее обновление: 30.10.2015



Важную роль в системе ООП играют интерфейсы. Они определяют некоторый абстрактный функционал, не имеющий конкретной реализации, который уже реализуют классы, наследующие эти интерфейсы. Определение интерфейса похоже на определение класса: интерфейс также может содержать свойства, методы и события. Чтобы объявить интерфейс, надо использовать ключевое слово **Interface** (обратите внимание, что имена интерфейсов обычно начинаются с заглавной буквы **I**). Итак, в прошлой главе мы создали небольшую систему из классов `Person`, `Employee` и `Client`, которые сейчас выглядят так:

```
1 Public MustInherit Class Person
2
3     Public Property FirstName() As String
4     Public Property LastName() As String
5     'Абстрактный метод
6     Public MustOverride Sub Display()
7
8     Public Sub New(fName As String, lName As String)
9         FirstName = fName
10        LastName = lName
11    End Sub
12
13 End Class
14
15 Public Class Employee
16     Inherits Person
17
18     Public Property Bank As String
19
20     Public Overrides Sub Display()
21         Console.WriteLine(FirstName & " " & LastName & " works in " & Bank)
22     End Sub
23
24     Public Sub New(fName As String, lName As String, _bank As String)
25         MyBase.New(fName, lName)
26         Bank = _bank
```

```

27     End Sub
28
29 End Class
30
31 Public Class Client
32     Inherits Person
33
34     Public Property Bank As String
35
36     Public Overrides Sub Display()
37         Console.WriteLine(FirstName & " " & LastName & " has an account in b
38     End Sub
39
40     Public Sub New(fName As String, lName As String, _bank As String)
41         MyBase.New(fName, lName)
42         Bank = _bank
43     End Sub
44
45 End Class

```

Теперь добавим в наше приложение интерфейс **IAccount**, который будет содержать методы и свойства, которые понадобятся при работе с счетом клиента. Чтобы добавить интерфейс, в меню **Project** выберите пункт **Add New Item...** и в появившемся списке выберите пункт **Code File**. Назовите новый файл **IAccount.vb**. Будет создан пустой файл, и затем добавьте в него следующий код интерфейса:

```

1  Public Interface IAccount
2      'Текущая сумма на счете
3      ReadOnly Property CurentSum() As Integer
4      'Метод для добавления денег на счет
5      Sub Put(sum As Integer)
6      'Метод для снятия денег со счета
7      Sub Withdraw(sum As Integer)
8      'Процент начислений
9      ReadOnly Property Procentage() As Integer
10 End Interface

```

Обратите внимание, что методы и свойства не имеют реализации, в этом они сближаются с абстрактными методами абстрактных классов. Сущность нашего интерфейса проста: он определяет два свойства для текущей суммы денег на счете и ставки процента по вкладам и два метода для добавления денег на счет и изъятия денег. Теперь нам надо реализовать интерфейс в классе **Client**, так как клиент у нас обладает счетом. Чтобы реализовать интерфейс, нам надо использовать ключевое слово **Implements**. Изменим класс **Client** следующим образом:

```

1  Public Class Client
2      Inherits Person
3      Implements IAccount

```

```

4
5 'Переменная для хранения суммы
6 Dim _sum As Integer
7 'Переменная для хранения процента
8 Dim _procentage As Integer
9
10 Public Property Bank As String
11
12 'Текущая сумма на счете
13 ReadOnly Property CurentSum() As Integer Implements IAccount.CurentSum
14     Get
15         Return _sum
16     End Get
17 End Property
18 'Метод для добавления денег на счет
19 Sub Put(sum As Integer) Implements IAccount.Put
20     _sum += sum
21 End Sub
22 'Метод для снятия денег со счета
23 Sub Withdraw(sum As Integer) Implements IAccount.Withdraw
24     If sum <= CurentSum Then
25         _sum -= sum
26     End If
27 End Sub
28 'Процент начислений
29 ReadOnly Property Procentage() As Integer Implements IAccount.Procentage
30     Get
31         Return _procentage
32     End Get
33 End Property
34
35 Public Overrides Sub Display()
36     Console.WriteLine(FirstName & " " & LastName & " has an account in bank " & Bank)
37 End Sub
38
39 Public Sub New(fName As String, lName As String, _bank As String, _sum As Integer)
40     MyBase.New(fName, lName)
41     Bank = _bank
42     Me._sum = _sum
43 End Sub

```

Обратите внимание, что класс, реализующий интерфейс, обязан реализовать все его свойства, методы и события. Интерфейсы, как и классы, могут наследоваться:

```

1 Public Interface IDepositAccount
2     Inherits IAccount
3
4     'Начисление процентов
5     Sub GetIncome()

```

6

7 [End Interface](#)

Зачем же нужны интерфейсы, если по сути они ничего не делают, только объявляют методы и свойства? Во-первых, интерфейсы позволяют реализовать концепцию множественного наследования. Если некоторый класс может иметь только один базовый класс, то при этом он может реализовать множество интерфейсов. Во-вторых, они более гибки по сравнению с классами, так как не содержат конкретной реализации.

[Назад](#) [Содержание](#) [Вперед](#)



ТАКЖЕ НА METANIT.COM

Ассемблер MASM. Установка и начало ...

3 месяца назад · 4 коммент...

Ассемблер MASM.
Установка и начало
работы, Visual Studio, ...

ListView

2 месяца назад · 1 коммент...

ListView в JavaFX,
создание списков,
получение выбранных в ...

Встроенные компоненты ввода

5 месяцев назад · 1 коммен...

Встроенные компоненты
ввода Blazor из
пространства имен ...

Взаим код

5 меся

Взаим
Pythor
языке

0 Комментариев

1

 Войти ▼

G

Начать обсуждение...

ВОЙТИ С ПОМОЩЬЮ

ИЛИ ЧЕРЕЗ DISQUS

?

Имя



Поделиться

Лучшие Новые Старые

Прокомментируйте первым.

Подписаться

О защите персональных данных

Помощь сайту

YooMoney:
410011174743222

Qiwi:
qiwi.com/n/METANIT

Перевод на карту
Номер карты:
4048415020898850

[Вконтакте](#) | [Телеграм](#) | [Twitter](#) | [Помощь сайту](#)

Контакты для связи: metanit22@mail.ru

Copyright © metanit.com, 2023. Все права защищены.