

Разработка приложений на платформе .NET

Лекция 4

Интерфейсы

Задание

- Реализуйте очередь в виде списка, содержащую комплексные числа
- Реализуйте методы
 - `void Enqueue(Complex c)` – помещает число в очередь (в конец)
 - `Complex Dequeue()` – получает число из начала очереди и удаляет его из очереди
 - `Complex Peek()` – возвращает число, находящееся в начале очереди
 - `int Count()` – возвращает кол-во элементов в очереди
 - `void Print()` - метод, распечатывающий содержимое очереди.
- Где `Complex` – класс комплексных чисел, со свойствами `Re` и `Im` и переопределённым методом `ToString()`

Интерфейсы

- Интерфейс - контракт, поведение, которое реализующий класс обязуется поддерживать.
- Набор семантически связанных абстрактных членов.
- Интерфейс определяет спецификацию, но не определяет реализацию
- Класс – наследуют, а интерфейс – реализуют

Объявление интерфейса

Объявление интерфейса

```
[attributes] [modifiers] interface interface_name [:  
    base_interfaces]  
{  
    interface_body  
}
```

Пример объявления интерфейса

```
public interface IDisposable {  
    void Dispose();  
}
```

```
public interface IPointy {  
    byte Points {get;}  
    int GetNumberOfPoints();  
}
```

Члены интерфейса

- Интерфейс может содержать следующие члены
 - Методы
 - Свойства
 - Индексаторы
 - События
 - Не могут содержать поля
 - Не могут содержать реализацию
 - Члены интерфейса не могут быть статическими
 - Члены интерфейса не могут иметь модификаторов
- Члены интерфейса всегда **public**

Реализация интерфейса

- Классы и структуры могут реализовать интерфейсы
 - `public class Circle2d : Ellipse2d, ICloneable`
 - Базовый класс должен идти первым в списке, а за ним список реализуемых интерфейсов.
- Можно реализовать одновременно несколько интерфейсов
- Класс, реализующий интерфейс:
 - Либо реализует все его методы
 - Либо объявляет их (и себя) абстрактными

Явная и неявная реализация

● Неявная реализация интерфейса

- Записывается короткое (как обычно) имя каждого метода, свойства и т.д.

```
public object Clone() { ... }
```

- Доступность члена извне:

- Метод доступен как обычно

```
peremen.Clone()
```

- Метод доступен и через интерфейс

```
((ICloneable)peremen).Clone()
```

```
IDisposable classForDispose = (IDisposable)peremen;
```

```
classForDispose.Dispose()
```

● Явная (**explicit**) реализация интерфейса

- Записывается квалифицированное имя каждого метода, свойства и т.д. (т.е. с указанием интерфейса)

- **object ICloneable.Clone()** { ... }

- Метод доступен только через интерфейс

```
((ICloneable)peremen).Clone()
```

- Используется при наличии конфликта

- Не указывается модификатор доступа. Он неявно **public**

Привязка методов интерфейса

- ◎ Привязка к не виртуальным методам
 - Производному классу, чтобы сменить реализацию, требуется самому реализовать интерфейс
- ◎ Привязка к виртуальным методам
 - Производным классам достаточно просто переопределить виртуальный метод, чтобы сменить реализацию

Об интерфейсах

- Интерфейсы могут наследоваться от других интерфейсов.
 - Производный интерфейс включает в себя все из базовых интерфейсов
 - Тип (класс, структура), реализующие производный интерфейс должен реализовать также и все базовые интерфейсы
 - `public interface IList : ICollection, IEnumerable { ...}`
- Переменные интерфейсного типа
 - Могут принимать значения любого типа, реализующего интерфейс
 - `IEnumerabe sequence = new int[] {1,2,3,4,5}`
- Передача интерфейсов в качестве параметров
 - Метод принимает любой тип, реализующий данный интерфейс
 - `public void GetShape(IDrawable shape);`
- Интерфейсы в качестве возвращаемых параметров
 - Метод возвращает некий тип, который реализует данный интерфейс
 - `public IDrawable GetShape();`
- В **.NET** все существующие интерфейсы начинаются с **I**

Абстрактные классы и интерфейсы

- Абстрактные классы и интерфейсы обеспечивают абстракцию
- Абстрактные классы могут содержать реализацию, интерфейсы – нет
- Интерфейсы могут содержать только методы, свойства, события и индексаторы
- Все члены интерфейса – public. Это открытый контракт
- Используйте абстрактные классы, если:
 - Требуется определить определенную функциональность, которую могут использовать все наследники
 - Производный класс “является” базовым
- Используйте интерфейсы
 - Производный класс “реализует” интерфейс
 - Требуется множественное наследование

Примеры интерфейсов

- **ICloneable**
 - Определяет единственный метод `object Clone()`
 - Позволяет выполнять копирование объектов
- **IDisposable**
 - Определяет единственный метод `Dispose()`
 - Реализуйте этот метод, если требуется освобождение внешних ресурсов
 - Файлы, соединения и т.д.
- **IComparable** – используется при сортировках при сравнении 2 элементов
- **IEnumerable, IEnumerable<T>, IEnumerator, IEnumerator<T>**
 - Позволяют итерировать по коллекции
- **IDictionary<TKey, TValue>, IList<T>, ICollection<T>**
 - Интерфейсы реализуемые коллекциями

Демонстрации

Интерфейсы

Задание по желанию

- Реализуйте абстрактный класс **Shape**, содержащий метод **Draw()**, якобы рисующий фигуру (вывод строки на экран)
- Создайте классы его потомков **Triangle**, **Hexagon**, **Circle**, **Sphere**
- Создайте интерфейсы
 - **IPoint**, со свойством **Point**, выдающим количество точек в фигуре.
 - **IDrawable**, с методом **Draw()**, якобы рисующем фигуру (вывод строки на экран)
- Реализуйте
 - **IPoint** для **Triangle** и **Hexagon**
 - **IDrawable** для **Triangle** (не явно) и **Sphere** (явно)
- В основном классе:
 - Создайте метод **AnalyzeShape()**, принимающий **Shape** и распечатывающий, если возможно, кол-во точек в фигуре, и вызывающий метод **Draw** напрямую и через интерфейс.
 - В методе **Main** создайте массив из **Shape** с разными фигурами и проанализируйте их вызвав метод **AnalyzeShape()**