Абстракция

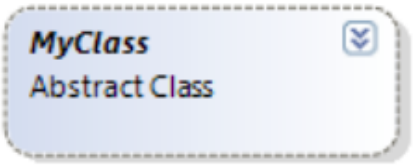
Абстракция в ООП–это придание объекту характеристик, которые отличают его от всех других объектов, четко определяя его концептуальные границы.

Абстрагирование в ООП – это способ выделить набор значимых характеристик объекта, исключая из рассмотрения незначимые. Соответственно, абстракция—это набор всех таких характеристик.

Абстрактные классы

Абстрактный класс в объектно-ориентированном программировании — это базовый класс, который не предполагает создания экземпляров через вызов конструктора напрямую, но экземпляр абстрактного класса создается неявно при построении экземпляра производного конкретного класса.

Абстрактные классы реализуют на практике один из принципов ООП - полиморфизма. Абстрактный класс может содержать (и не содержать) абстрактные методы и свойства. Абстрактный метод не реализуется для класса, в котором описан, однако должен быть реализован для его неабстрактных потомков. Абстрактные классы представляют собой наиболее общие абстракции, то есть имеющие наибольший объем и наименьшее содержание.




```
abstract class MyClass
{

}
```

Ключевое слово `abstract`может использоваться с ***классами, методами, свойствами, индексаторами и событиями***

Возможности и ограничения абстрактных классов

- Экземпляр абстрактного класса создать нельзя через вызов конструктора напрямую, но экземпляр абстрактного класса создается неявно при построении экземпляра производного конкретного класса
- Абстрактные классы могут содержать как абстрактные, так и не абстрактные члены
- Неабстрактный (конкретный) класс, являющийся производным от абстрактного, должен содержать фактические реализации всех наследуемых абстрактных членов

Абстрактные методы

Возможности абстрактных методов

- Абстрактный метод является неявным виртуальным методом

- Создание абстрактных методов допускается только в абстрактных классах
- Тело абстрактного метода отсутствует; создание метода просто заканчивается двоеточием, а после сигнатуры ставить фигурные скобки ({ }) не нужно
- Реализация предоставляется методом переопределения override, который является членом неабстрактного класса
- Абстрактный класс должен предоставлять реализацию для всех членов интерфейса.
- Абстрактный класс, реализующий интерфейс, может отображать методы интерфейса в абстрактных методах

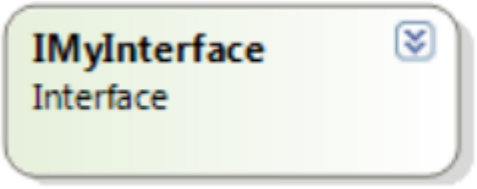
Интерфейсы

Интерфейс — семантическая и синтаксическая конструкция в коде программы, используемая для специфицирования услуг, предоставляемых классом или компонентом.

Интерфейс -стереотип, являющийся аналогом чистого абстрактного класса, в котором запрещена любая реализация

Для имени интерфейса следует применять в качестве префикса букву "`I`".

Это подсказывает, что данный тип является интерфейсом.




```
interface IMyInterface
{

}
```

Правила использования интерфейсов:

- Невозможно создать экземпляр интерфейса.
- Интерфейсы и члены интерфейсов являются абстрактными. Интерфейсы не имеют реализациив C#.
- Интерфейс может содержать только абстрактные члены (методы, свойства, события или индексаторы).
- Члены интерфейсов автоматически являются открытыми, абстрактными, и они не могут иметь модификаторов доступа.
- Интерфейсы не могут содержать константы, поля, операторы, конструкторы экземпляров, деструкторы или вложенные типы (интерфейсы в том числе).
- Класс или структура, которые реализуют интерфейс, должны реализовать члены этого интерфейса, указанные при его создании.
- Интерфейс может наследоваться от одного или нескольких базовых интерфейсов.
- Базовый класс так же может реализовать члены интерфейса с помощью виртуальных членов. В этом случае производный класс может изменить поведение интерфейса путем переопределения виртуальных членов.
- Если класс реализует два интерфейса, содержащих член с одинаковой сигнатурой, то при реализации этого члена в классе оба интерфейса будут использовать этот член для своей реализации.
- Если члены двух интерфейсов с одинаковой сигнатурой методов должны выполнять различные действия при их реализации, необходимо воспользоваться явной реализацией

члена интерфейса — техникой явного указания в имени члена имени интерфейса, которому принадлежит данный член. Это достигается путем включения в имя члена класса имени интерфейса с точкой. Данный член в производном классе будет помечен по умолчанию как скрытый.

Преимущество использования интерфейсов:

- Класс или структура может реализовать несколько интерфейсов (допустимо множественное наследование от интерфейсов).
- Если класс или структура реализует интерфейс, она получает только имена и сигнатуры метода.
- Интерфейсы определяют поведение экземпляров производных классов.
- Базовый класс может обладать не нужным функционалом, полученным от других его базовых классов, чего можно избежать, применяя интерфейсы.

Паттерн внедрение зависимости

Внедрение зависимостей — методика для создания слабо связанных приложений. Она предоставляет возможности для упрощения кода, извлечения и обработки зависимостей между объектами и автоматического создания экземпляров зависимого объекта.

Использование внедрения зависимостей предоставляет несколько преимуществ:

- Ослабление связи между классами
- Создание кода, который лучше поддается проверке.
- Упрощение тестирования.