

Абстрактные классы и интерфейсы

№ 4 Курс: C# Essential
урока:

Средства обучения: Компьютер с установленной Visual Studio

Обзор, цель и назначение урока

Рассмотрение абстрактных классов.

Рассмотрение интерфейсов.

Изучив материал данного занятия, учащийся сможет:

- Использовать и создавать абстрактные классы.
- Использовать и создавать интерфейсы.

Содержание урока

- Абстрактные классы.
- Интерфейсы.
- Паттерн внедрение зависимости

Резюме

- **Абстракция** – в объектно-ориентированном программировании это придание объекту характеристик, которые отличают его от всех других объектов, четко определяя его концептуальные границы. Основная идея состоит в том, чтобы отделить способ использования составных объектов данных от деталей их реализации, в виде более простых объектов, подобно тому, как функциональная абстракция разделяет способ использования функции и деталей её реализации, в терминах более примитивных функций, таким образом, данные обрабатываются функцией высокого уровня с помощью вызова функций низкого уровня.
- **Абстрактный класс** в объектно-ориентированном программировании – базовый класс, который не предполагает создания экземпляров. 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.
- Ключевое слово **abstract** может использоваться с **классами, методами, свойствами, индексаторами и событиями**.
- Ключевое слово **abstract** при создании класса указывает, что класс предназначен только для использования в качестве базового класса для других классов. Члены, помеченные как абстрактные или включенные в абстрактный класс, должны быть реализованы с помощью классов, производных от абстрактных классов.
- Возможности абстрактных классов:
 - Экземпляр абстрактного класса создать нельзя
 - Абстрактные классы могут содержать как абстрактные, так и обыкновенные (неабстрактные) члены.
 - Неабстрактный класс, являющийся производным от абстрактного, должен содержать фактические реализации всех наследуемых абстрактных методов и методов доступа.
- 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 точкой с запятой, 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.
- Преимущества использования абстрактных классов:
 - Общий код в одной реализации в виде конкретных и абстрактных членов
 - Изменение значения полей или неабстрактных членов абстрактного класса приводит к соответствующему изменению во всех его производных классах.
 - Наличие реализации по умолчанию.
- Интерфейс (от лат. *inter* – «между», и *face* – «поверхность») – семантическая и синтаксическая конструкция в коде программы, используемая для специфицирования услуг, предоставляемых классом или компонентом. Интерфейс определяет границу взаимодействия между классами или компонентами, специфицируя определенную абстракцию, которую осуществляет реализующая сторона. В отличие от многих других видов интерфейсов, интерфейс в ООП является строго формализованным элементом объектно-ориентированного языка и, в качестве семантической конструкции, широко используется кодом программы.
- Невозможно создать экземпляр интерфейса.
- Интерфейсы и члены интерфейсов являются абстрактными. Интерфейсы не имеют реализации по умолчанию.
- Интерфейс содержит только сигнатуры **методов, свойств, событий или индексаторов**.
- **Члены интерфейсов** автоматически являются **открытыми, абстрактными**, и они **не могут иметь модификаторы доступа**.
- Интерфейсы не могут содержать константы, поля, операторы, конструкторы экземпляров, деструкторы или типы.
- Класс или структура, которые реализуют интерфейс, должны реализовать члены этого интерфейса, указанные при его создании.
- Однако если базовый класс реализует интерфейс, производный также класс наследует эту реализацию.
- Интерфейс может наследоваться от одного или нескольких базовых интерфейсов.
- Базовый класс также 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.
 - Если класс или структура реализует интерфейс, она получает только имена и сигнатуры метода
 - Интерфейсы определяют поведение экземпляров производных классов
 - Базовый класс может обладать ненужным функционалом, полученным от других его базовых классов, чего можно избежать, применяя интерфейсы
- Внедрение зависимостей (Dependency Injection, DI) – превосходная методика для создания слабосвязанных приложений. Она предоставляет возможности для упрощения кода, извлечения и обработки зависимостей между объектами и автоматического создания экземпляров зависимого объекта.
- Внедрение зависимостей описывает процесс разработки приложений – вместо указания конкретных зависимостей в приложении во время разработки и создания необходимых объектов в коде во время выполнения приложение решает, какие объекты ему требуются, а потом создает и внедряет их в приложение.
- Использование внедрения зависимостей предоставляет несколько преимуществ:

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

Закрепление материала

- Что такое абстрактный класс?
- Что такое интерфейс?
- Чем абстрактный класс отличается от интерфейса?
- Что такое множественное наследование?
- Чем абстрактный класс отличается от конкретного?
- Какие члены могут быть абстрактными?

Дополнительное задание

Задание

Используя Visual Studio, создайте проект по шаблону Console Application.

Требуется: Изменить 12 пример первого урока (работа с документом) и создать общий абстрактный класс для всех частей документа.

Самостоятельная деятельность учащегося

Задание 1

Выучите основные конструкции и понятия, рассмотренные на уроке.

Задание 2

Используя Visual Studio, создайте проект по шаблону Console Application.

Требуется:

Создайте класс `AbstractHandler`.

В теле класса создать методы `void Open()`, `void Create()`, `void Change()`, `void Save()`.

Создать производные классы `XMLHandler`, `TXTHandler`, `DOCHandler` от базового класса `AbstractHandler`.

Написать программу, которая будет выполнять определение документа и для каждого формата должны быть методы открытия, создания, редактирования, сохранения определенного формата документа.

Задание 3

Используя Visual Studio, создайте проект по шаблону Console Application.

Требуется:

Создайте 2 интерфейса `IPlayable` и `IRecordable`. В каждом из интерфейсов создайте по 3 метода `void Play()` / `void Pause()` / `void Stop()` и `void Record()` / `void Pause()` / `void Stop()` соответственно.

Создайте производный класс `Player` от базовых интерфейсов `IPlayable` и `IRecordable`.

Написать программу, которая выполняет проигрывание и запись.

Задание 4

Зайдите на сайт MSDN.

Используя поисковые механизмы MSDN, найдите самостоятельно описание темы по каждому примеру, который был рассмотрен на уроке, так, как это представлено ниже, в разделе «Рекомендуемые ресурсы», описания данного урока. Сохраните ссылки и дайте им короткое описание.

Рекомендуемые ресурсы

MSDN: Абстрактные классы `abstract` (Руководство по программированию на C#)
<http://msdn.microsoft.com/ru-ru/library/sf985hc5.aspx>

MSDN: Интерфейсы (Руководство по программированию на C#)
<http://msdn.microsoft.com/ru-ru/library/ms173156.aspx>

MSDN: Интерфейс `interface` (Справочник по C#)
<http://msdn.microsoft.com/ru-ru/library/87d83y5b.aspx>