

Наследование и полиморфизм

№ урока: 3 Курс: C# Essential

Средства обучения: Компьютер с установленной Visual Studio

Обзор, цель и назначение урока

Рассмотрение и применение модификаторов доступа.
Рассмотрение понятия инкапсуляции и механизмов наследования.
Рассмотрение полиморфизма.

Изучив материал данного занятия, учащийся сможет:

- Понимать работу наследования.
- Применять различные модификаторы доступа.
- Отличать и применять основные формы полиморфизма

Содержание урока

1. Рассмотрение понятия наследования.
2. Обзор и применение модификаторов доступа.
3. Вызов конструктора базового класса.
4. Приведение к базовому типу.
5. Понятие Upcast-а и DownCast-а.
6. Рассмотрение понятия полиморфизма.
1. Операторы **Is** и **As**.
2. Cast с использованием оператора **as**
7. Использование герметизированных классов.

Резюме

- ООП – Объектно-ориентированное программирование – парадигма программирования, в которой основными концепциями являются понятия объектов и классов.
- Наследование – механизм объектно-ориентированного программирования (наряду с инкапсуляцией, полиморфизмом и абстракцией), позволяющий описать новый класс на основе уже существующего (родительского), при этом свойства и функциональность родительского класса заимствуются новым классом.
- Недостаток наследования – хрупкий базовый класс. Хрупкий базовый класс – фундаментальная проблема объектно-ориентированного программирования. Проблема хрупкого базового класса заключается в том, что малейшие правки в деталях реализации базового класса могут привести ошибку в производные классы. В худшем случае это приводит к тому, что любая успешная модификация базового класса требует предварительного изучения всего дерева наследования, и зачастую невозможна (без создания ошибок) даже в этом случае.
- Рекомендуется использовать следующие пары:
Базовый класс – Производный класс
Супер класс – Подкласс или (сабкласс)
Родительский класс – Дочерний класс
Класс Родитель – Класс Потомок
- Модификаторы доступа – это ключевые слова, задающие объявленную доступность члена или типа. При помощи модификаторов доступа можно задать следующие пять уровней доступности:
 - 1) **public** – доступ к типу или члену возможен из любого другого кода в той же сборке или другой сборке, ссылающейся на него.
 - 2) **protected** – доступ к типу или элементу можно получить только из кода в том же классе, либо в производном классе.

- 3) **internal** – доступ к типу или члену возможен из любого кода в той же сборке, но не из другой сборки.
 - 4) **protected internal** – доступ ограничен текущей сборкой или типами, которые являются производными от содержащего класса.
 - 5) **private** – доступ к типу или члену можно получить только из кода в том же классе или структуре.
- Полиморфизм – возможность объектов с одинаковой спецификацией иметь различную реализацию.
 - Полиморфизм предоставляет подклассу способ определения собственной версии метода, определенного в его базовом классе, с использованием процесса, который называется переопределением метода (method overriding).
 - Базовые классы могут определять и реализовывать виртуальные методы, а производные классы могут переопределять их. Это означает, что они предоставляют свои собственные определение и реализацию.
 - Во время выполнения, когда клиентский код вызывает метод, среда CLR ищет тип времени выполнения объекта и вызывает это переопределение виртуального метода. Таким образом, в исходном коде можно вызвать метод в базовом классе и вызвать выполнение метода с версией производного класса.
 - Если производный класс наследует от базового класса, то он приобретает все методы, поля, свойства и события базового класса. Проектировщик производного класса может выбирать из следующих возможностей:
 - 1) переопределить виртуальные члены в базовом классе
 - 2) наследовать метод последнего базового класса без его переопределения
 - 3) определить новую не виртуальную реализацию этих членов, которая скрывает реализации базового класса.
 - Поля не могут быть виртуальными.
 - Виртуальными могут быть только методы, свойства, события и индексаторы.
 - Если в производном классе виртуальный метод переопределяется, то этот член вызывается даже в том случае, если доступ к экземпляру этого класса осуществляется как к экземпляру базового класса.
 - Виртуальные методы и свойства дают возможность производным классам расширять базовый класс, без необходимости использования реализации метода базового класса.
 - Если необходимо, чтобы производный член имел то же имя, что и член базового класса, но не нужно, чтобы он участвовал в виртуальном вызове, можно использовать ключевое слово **new**. Ключевое слово **new** располагается перед возвращаемым типом замещаемого члена класса.
 - Оператор **is** - проверяет совместимость объекта с заданным типом.
 - Если предоставленный объект может быть приведен к предоставленному типу не вызывая исключение, выражение **is** принимает значение **true**.
 - Оператор **as** используется для выполнения преобразований между совместимыми ссылочными типами
 - Оператор **as** подобен оператору приведения. Однако, если преобразование невозможно, **as** возвращает значение **null**, а не вызывает исключение
 - В общем виде логика работы оператора **as** представляет собой механизм использования оператора **is**, только в сокращенном виде
 - Ключевое слово — **sealed**, которое предотвращает наследование. Если класс помечен как **sealed** (запечатанный), компилятор не позволяет наследовать от него. Считается, что класс герметизирован или «запечатан».

Закрепление материала

- Что такое наследование?
- Какие недостатки наследования вы знаете?
- Что такое модификаторы доступа и где их используют?
- Что такое ООП?

- Назовите основные парадигмы ООР.
- Что такое полиморфизм?
- Зачем используется ключевое слово `virtual`?
- Что такое Cast, Upcast, Downcast?
- Объясните назначение ключевого слова `sealed`?

Дополнительное задание

Задание

Используя Visual Studio, создайте проект по шаблону Console Application.

Требуется:

Создайте класс `Printer`.

В теле класса создайте метод `void Print(string value)`, который выводит на экран значение аргумента.

Реализуйте возможность того, чтобы в случае наследования от данного класса других классов, и вызове соответствующего метода их экземпляра, строки, переданные в качестве аргументов методов, выводились разными цветами.

Обязательно используйте приведение типов.

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:

Создать класс, представляющий учебный класс `ClassRoom`.

Создайте класс ученик `Pupil`. В теле класса создайте методы `void Study()`, `void Read()`, `void Write()`, `void Relax()`.

Создайте 3 производных класса `ExcelentPupil`, `GoodPupil`, `BadPupil` от класса базового класса `Pupil` и переопределите каждый из методов, в зависимости от успеваемости ученика.

Конструктор класса `ClassRoom` принимает аргументы типа `Pupil`, класс должен состоять из 4 учеников.

Предусмотрите возможность того, что пользователь может передать 2 или 3 аргумента.

Выведите информацию о том, как все ученики экземпляра класса `ClassRoom` умеют учиться, читать, писать, отдыхать.

Задание 3

Используя Visual Studio, создайте проект по шаблону Console Application.

Требуется:

Создать класс `Vehicle`.

В теле класса создайте поля: координаты и параметры средств передвижения (цена, скорость, год выпуска).

Создайте 3 производных класса `Plane`, `Car` и `Ship`.

Для класса `Plane` должна быть определена высота и количество пассажиров.

Для класса `Ship` — количество пассажиров и порт приписки.

Написать программу, которая выводит на экран информацию о каждом средстве передвижения.

Задание 4

Используя Visual Studio, создайте проект по шаблону Console Application.

Требуется:

Создайте класс `DocumentWorker`.

В теле класса создайте три метода `OpenDocument()`, `EditDocument()`, `SaveDocument()`.

В тело каждого из методов добавьте вывод на экран соответствующих строк: "Документ открыт", "Редактирование документа доступно в версии Про", "Сохранение документа доступно в версии Про".

Создайте производный класс `ProDocumentWorker`.

Переопределите соответствующие методы, при переопределении методов выводите следующие строки: "Документ отредактирован", "Документ сохранен в старом формате, сохранение в остальных форматах доступно в версии Эксперт".

Создайте производный класс `ExpertDocumentWorker` от базового класса `ProDocumentWorker`.

Переопределите соответствующий метод. При вызове данного метода необходимо выводить на экран "Документ сохранен в новом формате".

В теле метода `Main()` реализуйте возможность приема от пользователя номера ключа доступа **pro** и **exp**. Если пользователь не вводит ключ, он может пользоваться только бесплатной версией (создается экземпляр базового класса), если пользователь ввел номера ключа доступа **pro** и **exp**, то должен создаваться экземпляр соответствующей версии класса, приведенный к базовому – `DocumentWorker`.

Задание 5

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: Полиморфизм

[http://msdn.microsoft.com/ru-ru/library/z165t2xk\(v=VS.90\).aspx](http://msdn.microsoft.com/ru-ru/library/z165t2xk(v=VS.90).aspx)

MSDN: Модификаторы доступа (Справочник по C#)

[http://msdn.microsoft.com/ru-ru/library/wxh6fsc7\(v=VS.90\).aspx](http://msdn.microsoft.com/ru-ru/library/wxh6fsc7(v=VS.90).aspx)

MSDN: Наследование (Руководство по программированию на C#)

<http://msdn.microsoft.com/ru-ru/library/ms173149.aspx>