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.
7. Операторы Is и As.
8. 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 привнести 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 или структуре, 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 OOP.
* Что такое полиморфизм?
* Зачем используется ключевое слово virtual?
* Что такое Cast, Upcast, Downcast?
* Объясните назначение ключевого слова sealed?