

Классы и объекты

№ урока: 1 Курс: C# Essential

Средства обучения: Компьютер с установленной Visual Studio

Обзор, цель и назначение урока

Рассмотрение классов и работа с экземплярами классов.

Рассмотрение основных членов класса.

Изучение назначения свойств и конструкторов.

Изучив материал данного занятия, учащийся сможет:

- Понимать работу класса.
- Использовать и создавать свойства.
- Понимать назначение и применение конструкторов.

Содержание урока

1. Рассмотрение ООП.
2. Обзор классов.
3. Обзор объектов.
4. Создание и использование классов.
5. Члены класса.
6. Свойства.
7. ReadOnly и WriteOnly свойства.
8. Конструкторы.

Резюме

- **ООП - Объектно-ориентированное программирование** — парадигма программирования, в которой основными концепциями являются понятия объектов и классов.
- **Класс (лат. classis — группа)** — группа предметов или явлений, обладающих общими признаками.
- **Класс** — это конструкция языка, состоящая из ключевого слова `class`, идентификатора и тела. Класс может содержать в своем теле поля и методы. Также классы могут включать в свое тело другие классы, но такой подход не является широко распространённой техникой.
- **Объект** — это некоторая сущность, обладающая определённым состоянием и поведением, имеет заданные значения свойств (полей) и операций над ними (методов).
- Объект состоит из следующих частей:
 1. имя объекта;
 2. состояние (переменные состояния);
 3. методы (операции).
- **Свойство** — это способ доступа к внутреннему состоянию объекта, имитирующий переменную некоторого типа. Обращение к свойству объекта выглядит так же, как и обращение к структурному полю (в структурном программировании), но, в действительности, реализовано через вызов функции. При попытке задать значение данного свойства вызывается один метод, а при попытке получить значение данного свойства — другой.
- **Экземпляр класса (instance)** — это описание конкретного объекта в памяти.
- **Инстанцирование (instantiation)** — создание экземпляра класса. В отличие от слова «создание», применяется не к объекту, а к классу. То есть, говорят: «создать экземпляр класса или инстанцировать класс».
- **Конструктор класса (constructor, иногда сокращают ctor)** — специальный блок инструкций, вызываемый при создании объекта.

- Рекомендуется всегда создавать явно конструктор по умолчанию.
- Рекомендуется использовать то же самое название для параметров конструктора и поля или свойства, если параметры конструктора используются для того чтобы инициализировать поле или установить свойство.
- Рекомендуется минимизировать работу конструктора.
- Конструкторы бывают двух видов конструкторы типа и конструкторы экземпляра класса. Конструкторы типа являются статическими и выполняются средой CLR до использования типа. Конструкторы экземпляра класса работают тогда, когда создается экземпляр класса.
- Конструкторы типа не могут принимать параметры.
- Конструктор не бывает виртуальным (в смысле как [virtual](#)).
- Конструкторы экземпляра могут принимать параметры.
- Конструкторы экземпляра класса, которые не принимают параметров, называют конструкторами по умолчанию.
- Конструкторы экземпляра класса, которые принимают параметры, называют пользовательскими конструкторами.
- **CTS – Common Type System (Общая система типов)** – это спецификация, определяющая, как какой-либо тип должен быть определен для правильного выполнения средой .Net.
- Спецификация для CTS закреплена в стандарте ECMA 335, озаглавленном «Common Language Infrastructure (CLI) Partitions I to VI». CLI и CTS были разработаны корпорацией Microsoft, а .NET Framework — реализация этого стандарта.

Закрепление материала

- Что такое класс?
- Что такое объект?
- Что такое экземпляр класса?
- Что такое ООП?
- Назовите основные парадигмы ООР.
- Что такое инкапсуляция?
- Что такое свойство?
- Какие виды свойств бывают?
- Что такое модификаторы доступа и где их используют?
- Какие типы конструкторов вы знаете?
- Зачем использовать конструкторы по умолчанию?

Дополнительное задание

Задание

Используя Visual Studio, создайте проект по шаблону Console Application.

Требуется:

Создать класс с именем [Address](#).

В теле класса требуется создать поля: index, country, city, street, house, apartment. Для каждого поля, создать свойство с двумя методами доступа.

Создать экземпляр класса [Address](#).

В поля экземпляра записать информацию о почтовом адресе.

Выведите на экран значения полей, описывающих адрес.

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 с именем `Rectangle`.

В теле класса создать два поля, описывающие длины сторон `double side1, side2`.

Создать пользовательский конструктор `Rectangle(double side1, double side2)`, в теле которого поля `side1` и `side2` инициализируются значениями аргументов.

Создать два метода, вычисляющие площадь прямоугольника - `double AreaCalculator()` и периметр прямоугольника - `double PerimeterCalculator()`.

Создать два свойства `double Area` и `double Perimeter` с одним методом доступа `get`.

Написать программу, которая принимает от пользователя длины двух сторон прямоугольника и выводит на экран периметр и площадь.

Задание 3

Используя Visual Studio, создайте проект по шаблону Console Application.

Требуется:

Создать класс `Book`. Создать классы `Title`, `Author` и `Content`, каждый из которых должен содержать одно строковое поле и метод `void Show()`.

Реализуйте возможность добавления в книгу названия книги, имени автора и содержания.

Выведите на экран разными цветами при помощи метода `Show()` название книги, имя автора и содержание.

Задание 4

Используя Visual Studio, создайте проект по шаблону Console Application.

Требуется:

Создать классы `Point` и `Figure`.

Класс `Point` должен содержать два целочисленных поля и одно строковое поле.

Создать три свойства с одним методом доступа `get`.

Создать пользовательский конструктор, в теле которого проинициализируйте поля значениями аргументов. Класс `Figure` должен содержать конструкторы, которые принимают от 3-х до 5-ти аргументов типа `Point`.

Создать два метода: `double LengthSide(Point A, Point B)`, который рассчитывает длину стороны многоугольника; `void PerimeterCalculator()`, который рассчитывает периметр многоугольника.

Написать программу, которая выводит на экран название и периметр многоугольника.

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: Классы `class` (Руководство по программированию на C#)

<http://msdn.microsoft.com/ru-ru/library/x9afc042.aspx>

MSDN: Объекты (Руководство по программированию на C#)

<http://msdn.microsoft.com/ru-ru/library/ms173110.aspx>

MSDN: Использование конструкторов (руководство по программированию в C#)

[http://msdn.microsoft.com/ru-ru/library/ms173115\(VS.90\).aspx](http://msdn.microsoft.com/ru-ru/library/ms173115(VS.90).aspx)

MSDN: Автоматически реализуемые свойства (Руководство по программированию на C#)

<http://msdn.microsoft.com/ru-ru/library/bb384054.aspx>