

# C# Essential

Перегрузка операторов

# C# Essential

После урока обязательно



Повторите этот урок в видео формате на [ITVDN.com](http://itvdn.com)



Проверьте как Вы усвоили данный материал  
на [TestProvider.com](http://testprovider.com)

# Перегрузка операторов

# Тип

## Object

В C# все типы, предопределенные и пользовательские, ссылочные типы и типы значений, наследуют непосредственно или косвенно от **Object**.

Так как все классы в платформе **.NET Framework** являются производными класса **Object**, все методы, определенные в классе **Object**, доступны для всех объектов в системе.



Переменным типа **Object** можно назначать значения любых типов.

# Переопределение

## Методы класса Object

- **Equals** – данный метод поддерживает сравнение объектов.
- **Finalize** – данный метод выполняет операции очистки перед автоматической утилизацией объекта.
- **GetHashCode** – данный метод создает число, соответствующее значению объекта, обеспечивающего возможность использования хэш-таблицы. Чтобы переопределить данный метод необходимо переопределить и метод **Equals**
- **ToString** – создает понятную для пользователя строку текста, в которой описывается экземпляр класса.
- **MemberwiseClone** – создает "неполную" копию объекта. При этом копируются члены, но не объекты, на которые ссылаются эти члены.

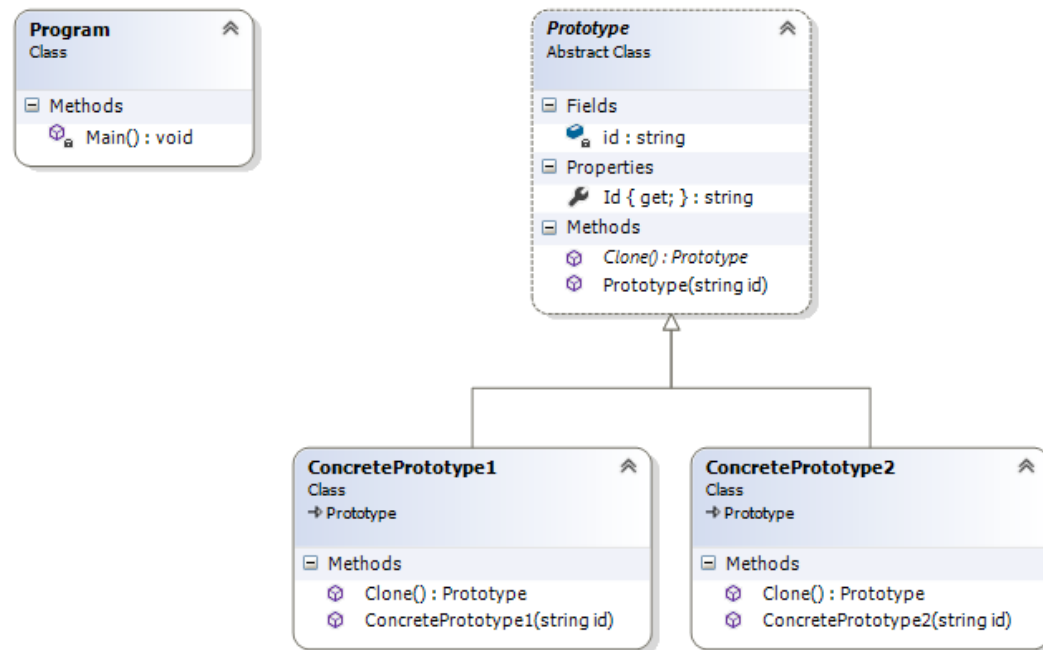


Клонирование ассоциации происходит поверхностно.  
Граф наследования копируется глубоко.

# Прототип

## Prototype

**Прототип** – паттерн, порождающий объекты. Он определяет, задает виды создаваемых объектов с помощью интерфейса некоторого экземпляра-прототипа, и создает новые объекты путем копирования (клонирования) этого экземпляра.



**Прототип** – это единственный паттерн из серии «порождающих паттернов», который для создания новых объектов использует не явное инстанцирование, а клонирование.

# ICloneable

## Интерфейс ICloneable

Интерфейс **ICloneable** поддерживает копирование, при котором создается новый экземпляр класса с тем же значением, что и у существующего экземпляра

```
public interface ICloneable
{
    object Clone();
}
```

Реализовав интерфейс **ICloneable**, можно создать все условия для копирования объекта. Интерфейс **ICloneable** содержит один член, **Clone**, предназначенный для поддержки копирования помимо выполняемого с помощью метода **MemberwiseClone**.

# Операторы

## Operators

**Оператор** – это элемент программы, который применяется к одному или нескольким операндам в выражении или операторе.

Операторы, получающие на вход один операнд, например, оператор инкремента (**++**) или **new**, называются унарными операторами.

Операторы, получающие на вход два операнда, например, арифметические операторы (**+**, **-**, **\***, **/**) называются бинарными.



# Перегрузка операторов

## operator

В C# пользовательские типы могут перегружать операторы путем определения функций статических членов с помощью ключевого слова `operator`.

```
public static Point operator +(Point p1, Point p2)
{
    return new Point(p1.x + p2.x, p1.y + p2.y);
}
```

```
static void Main()
{
    Point a = new Point(1, 1);
    Point b = new Point(2, 2);

    Point c = a + b;
}
```



Использовать ключевое слово `operator`, можно только вместе с ключевым словом `static`.

# Перегрузка операторов

## Правила перегрузок

Операторы сравнения можно перегружать, но только парами: если перегружен оператор `==`, то `!=` также должен быть перегружен.

Обратный принцип также действителен и действует для операторов `<` и `>`, а также для `<=` и `>=`.

Для перегрузки оператора в пользовательском классе нужно создать метод в классе с правильной сигнатурой.

Метод нужно назвать "`operator X`", где `X` – имя или символ перегружаемого оператора.

Унарные операторы имеют один параметр, а бинарные – два. В каждом случае один параметр должен быть такого же типа, как класс или структура, объявивший оператор.

# Оператор явного преобразования типа

explicit

```
public static explicit operator Digit(byte argument)
{
    Digit digit = new Digit(argument);
    return digit;
}
```

```
class MainClass
{
    static void Main()
    {
        byte variable = 1;

        // Явное преобразование byte-to-Digit.
        Digit digit = (Digit)variable;
    }
}
```

Ключевое слово **explicit** служит для создания оператора явного преобразования типа.

# Оператор неявного преобразования типа

implicit

```
public static implicit operator Digit(byte argument)
{
    Digit digit = new Digit(argument);
    return digit;
}
```

```
class MainClass
{
    static void Main()
    {
        byte variable = 1;

        // Неявное преобразование byte-to-Digit.
        Digit digit = variable;
    }
}
```

Ключевое слово **implicit** служит для создания оператора неявного преобразования типа.

# C# Essential

Q&A

# Информационный видеосервис для разработчиков программного обеспечения

