Как правило, доступ к члену класса

организуется посредством объекта этого класса, но в то же время можно создать

член класса для самостоятельного применения без ссылки на конкретный экземпляр

объекта. Для того чтобы создать такой член класса, достаточно указать в самом начале

его объявления ключевое слово static. Если член класса объявляется как static, то

он становится доступным до создания любых объектов своего класса и без ссылки на

какой-нибудь объект. С помощью ключевого слова static можно объявлять как переменные,

так и методы. Наиболее характерным примером члена типа static служит

метод Main(), который объявляется таковым потому, что он должен вызываться операционной

системой в самом начале выполняемой программы.

Для того чтобы воспользоваться членом типа static за пределами класса, достаточно

указать имя этого класса с оператором-точкой. Но создавать объект для этого не

нужно. В действительности член типа static оказывается доступным не по ссылке на

объект, а по имени своего класса. Так, если требуется присвоить значение 10 переменной

count типа static, являющейся членом класса Timer, то для этой цели можно

воспользоваться следующей строкой кода.

***Timer.count = 10;***

Эта форма записи подобна той, что используется для доступа к обычным переменным

экземпляра посредством объекта, но в ней указывается имя класса, а не объекта.

Аналогичным образом можно вызвать метод типа static, используя имя класса и

оператор-точку.

Когда же объекты объявляются в своем классе, то копия переменной типа static не

создается. Вместо этого все экземпляры класса совместно пользуются одной и той же

переменной типа static. Такая переменная инициализируется перед ее применением

в классе. Когда же ее инициализатор не указан явно, то она инициализируется

нулевым значением, если относится к числовому типу данных, пустым значением,

если относится к ссылочному типу, или же логическим значением false, если относится

к типу bool. Таким образом, переменные типа static всегда имеют какое-то

значение. (***glava8\_16***)

class StaticDemo

{

public static int Val = 100;

public static int ValDiv()

{

return Val / 2;

}

}

class SDemo

{

static void Main()

{

Console.WriteLine("Raw Variable "

+ "StaticDemo.Val = " + StaticDemo.Val);

StaticDemo.Val = 8;

Console.WriteLine("Curent Variable "

+ "StaticDemo.Val = " + StaticDemo.Val);

Console.WriteLine("StaticDemo.Valdiv(): "

+ StaticDemo.ValDiv());

}

}

Как следует из приведенного выше результата, переменная типа static инициализируется

до создания любого объекта ее класса.

На применение методов типа static накладывается ряд следующих ограничений.

• В методе типа static должна отсутствовать ссылка this, поскольку такой метод

не выполняется относительно какого-либо объекта.

• В методе типа static допускается непосредственный вызов только других методов

типа static, но не метода экземпляра из того самого же класса. Дело в

том, что методы экземпляра оперируют конкретными объектами, а метод типа

static не вызывается для объекта. Следовательно, у такого метода отсутствуют

объекты, которыми он мог бы оперировать.

• Аналогичные ограничения накладываются на данные типа static. Для метода

типа static непосредственно доступными оказываются только другие данные

типа static, определенные в его классе. Он, в частности, не может оперировать

переменной экземпляра своего класса, поскольку у него отсутствуют объекты,

которыми он мог бы оперировать.

Короче, если имеется тип ***static*** то он может обращаться ***только*** к ***static*** объектам, будь то функции либо переменный, ***static*** ***только*** к ***static***.

Следует особо подчеркнуть, что из метода типа static нельзя вызывать методы

экземпляра и получать доступ к переменным экземпляра его класса, как это

обычно делается посредством объектов данного класса. И объясняется это тем, что

без указания конкретного объекта переменная или метод экземпляра оказываются

недоступными. Например, приведенный ниже фрагмент кода считается совершенно

верным.

***class MyClass {***

***// Нестатический метод.***

***void NonStaticMeth() {***

***Console.WriteLine("В методе NonStaticMeth().");***

***}***

***/\* Нестатический метод может быть вызван из***

***статического метода по ссылке на объект. \*/***

***public static void staticMeth(MyClass ob) {***

***ob.NonStaticMeth(); // все верно!***

***}***

***}***

В данном примере метод NonStaticMeth() вызывается из метода staticMeth()

по ***ссылке*** на ***объект*** ob типа MyClass.

Поля типа static не зависят от конкретного объекта, и поэтому они удобны для

хранения информации, применимой ко всему классу. Ниже приведен пример программы,

демонстрирующей подобную ситуацию. В этой программе поле типа static

служит для хранения количества существующих объектов. (***glava8\_16.1***)

class CountInst

{

static int count = 0;

//increment when ob created

public CountInst()

{

count++;

}

//Decrement count

~CountInst()

{

count--;

}

public static int GetCount()

{

return count;

}

}

class CountDemo

{

static void Main()

{

CountInst ob;

for(int i =0;i<10;i++)

{

ob = new CountInst();

Console.WriteLine("Counter: " + CountInst.GetCount());

}

}

}

Всякий раз, когда создается объект типа CountInst, инкрементируется поле count

типа static. Но всякий раз, когда такой объект утилизируется, поле count декрементируется.

Следовательно, поле count всегда содержит количество существующих

в настоящий момент объектов. И это становится возможным только благодаря использованию

поля типа static.