Управление доступом в языке C# организуется с помощью четырех *модификаторов*

*доступа:* ***public***, ***private***, ***protected*** и internal. Модификатор ***protected*** применяется

только в тех случаях, которые связаны с наследованием. А модификатор ***internal*** служит в основном для ***сборки****,* которая в широком смысле означает в C# разворачиваемую программу или библиотеку.

Когда член класса обозначается спецификатором public, он становится доступным

из любого другого кода в программе, включая и методы, определенные в других

классах. Когда же член класса обозначается спецификатором private, он может быть

доступен только другим членам этого класса. Следовательно, методы из других классов

не имеют доступа к закрытому члену (private) данного класса. Если ни один из спецификаторов доступа не указан, член класса считается закрытым

для своего класса по умолчанию. Поэтому при создании закрытых членов класса

спецификатор private указывать для них необязательно.

Спецификатор доступа указывается перед остальной частью описания типа отдельного

члена. Это означает, что именно с него должен начинаться оператор объявления

члена класса.

***public string errMsg;***

***private double bal;***

***private bool isError(byte status) { // ...***

Подробнее в ***glava8\_1.***

class MyClass

{

private int alpha;

int beta; //also private(default)

public int gamma;

//members of class have access to private members

public void SetAlpha(int a)

{

alpha = a;

}

public int GetAlpha()

{

return alpha;

}

public void SetBeta(int a)

{

beta = a;

}

public int GetBeta()

{

return beta;

}

}

static void Main()

{

MyClass ob = new MyClass();

//access to privet elements alpha / beta

ob.SetAlpha(-99);

ob.SetBeta(19);

Console.WriteLine("ob.alpha = " + ob.GetAlpha());

Console.WriteLine("ob.beta = " + ob.GetBeta());

ob.gamma = 99; //gamma is public, so you can use it outside}

Короче говоря все как в С++. Если ***public*** – то можно обращаться к члену объекта класса из вне, тобишь в Main(). Если ***private*** – то можно обращаться к нему используя только ***public*** методы.

ориентированном программировании. И хотя для этого не существует твердо установленных

правил, ниже перечислен ряд общих принципов, которые могут служить

в качестве руководства к действию.

• Члены, используемые только в классе, должны быть закрытыми.

• Данные экземпляра, не выходящие за определенные пределы значений, должны

быть закрытыми, а при организации доступа к ним с помощью открытых методов

следует выполнять проверку диапазона представления чисел.

• Если изменение члена приводит к последствиям, распространяющимся за пределы

области действия самого члена, т.е. оказывает влияние на другие аспекты объекта,

то этот член должен быть закрытым, а доступ к нему — контролируемым.

• Члены, способные нанести вред объекту, если они используются неправильно,

должны быть закрытыми. Доступ к этим членам следует организовать с помощью

открытых методов, исключающих неправильное их использование.

• Методы, получающие и устанавливающие значения закрытых данных, должны

быть открытыми.

• Переменные экземпляра допускается делать открытыми лишь в том случае,

если нет никаких оснований для того, чтобы они были закрытыми.

Разумеется, существует немало ситуаций, на которые приведенные выше принципы

не распространяются, а в особых случаях один или несколько этих принципов

могут вообще нарушаться. Но в целом, следуя этим правилам, вы сможете создавать

объекты, устойчивые к попыткам неправильного их использования.