Классы и методы в C#

Зачем нужны классы?

Класс – это составной тип данных или некий шаблон, в котором кроме данных, так же описываются методы обработки и функции.

Основной данные в классе – это объекты:

Объекты обладают уникальностью и у каждого есть свое состояние и поведение.

Так записываются классы в c#:

[Модификатор доступа]class[имя\_класса]

{

//тело класса

}

Модификатор доступа к классам:

1. Publick - доступ к классу возможен из любого места одной сборки либо из другой сборки на которую есть ссылка.
2. Internal – доступ к классу возможен только из сборки в которой он объявлен.

Если при объявлении класса не указать модификатор доступа то по умолчанию будет применяться режим internal.

Сборка это готовый функциональный модуль в виде exe либо dll файла (файлов), который содержит скомпилированный код для .NET.

Объявление классов

Класс следует объявлять внутри пространства имен namespase, но за пределами другого класса

Пример:

Namespace HelloWorld

{

Class student // класс будет internal

{

//тело класса

}

Publick

}

Члены класса:

1. Поля – (класс студентов)
2. Константы
3. Свойства
4. Конструкторы
5. Методы (или функции)
6. События
7. Операторы
8. Индексаторы
9. Вложенные типы

Все члены класса так же имеют свой уровень доступа.

Модификаторы доступа к членам класса

1. Publick
2. Protected – доступ к члену возможен только внутри класса либо в классе – насследнике.
3. Internal
4. Private – доступ к члену возможен только внутри класса.
5. Protected internal – доступ к члену возможен из одной сборки либо ил класса наследника другой сборки.

Инкапсуляция – это использование модификаторов доступа с целью сокрытия частей программного кода от конечного пользователя (разработчика или наследующий объект)

Классы и члены класса можно сделать статическими с помозью служебного слова static

Publick static class class1

{

}

При использовании статического класса не обязательно создавать объект класса для обращения к классу и его членам.

Поля класса

Поле – переменная объявленная внутри класса. Для получения доступа к полям следует использоваться свойства или методы .

Class student

{

Private string firstName;

Private string lastName;

Private int age;

Public string group; // не рекомендуется использовать public для поля

}

Названия полей должна отображать за что они отвечают и они печатаются с маленькой буквы

Создание объектов

После объявления класса можно создавать объекты при помощи ключевого слова new и имени класса

Namespace HelloWorld

{

Class student

{

Private string firstName;

Private string lastName;

Private int age;

Public string group; // не рекомендуется использовать public для поля

}

Class program

}

Static void Main (string[]args)

{

}

Доступ к членам объекта

Доступ осуществляется при помощи оператора точка «.»

Константы

Константы – члены класса не отличаются от простых констант.

Константа это переменная значения которой нельзя изменить. Константа объявляется с помощью ключевого слова const.

Class math

{

Private const double Pi = 3.14;

}

Интерфейс

Интерфейс в отличии от класса описывает поведение объекта (какие действия выполнены над объектом).

Методы

Метод – это небольшая подпрограмма которая выполняет в идеале только одну функцию.

Методы позволяют сократить объем кода.

Различают статические и простые методы.

Статический метод – это метод который не имеет доступа к полям объекта и для вызова такого метода не нужно создавать экземпляр (объект) класса в котором он объявлен.

Простой метод – это метод, который имеет подступ к данным объекта и его вызов выполняется через объект.

Пример

Public double Meth1(double a)

{…

Rerun(Math.Pow(a,2))

}

2)

Private void Meth2()

{…

}

Свойства

Свойства класса – это член класса он предоставляет удобный механизм доступа к полю класса (чтения поля и запись)

При использовании свойства мы обращаемся к нему как к полю класса но на самом деле компилятор преобразовывает это обращение к вызову соответствующего неявного метода

Такой метод называется аксессор.

<тип><имя>

{

Get

{// код аксессора для чтения из поля }

Set

{// код аксессора для записи в поле }

}

Имя пишется с большой буквы, поле H имя h и так везде.

Get (для получения данных)

Set (для записи данных)

Метод доступа get должен заканчиваться оператором return or throw, а элемент уппавления не должен выходить за основную часть метода доступа.

События

Прежде чем рассмотреть события необходимо ознакомиться с понятием делегат.

Делегат – это объект указывающий на другой метод или список методов приложения который может быть вызван позднее.

Тип делегата объявляется с помощью ключевого слова delegate

Delegate<возвращаемый\_тип> имя

(список\_параметров)

Событие это ситуация при возникновении которой произойдет действие или несколько действий.

Событыие задается ключенвым словом event:

Public event <НазваниеДелегата><НазваниеСобытия>;