C#高级编程：对象和类型 - 文章 - 伯乐在线

原文出处： Enstein_jun

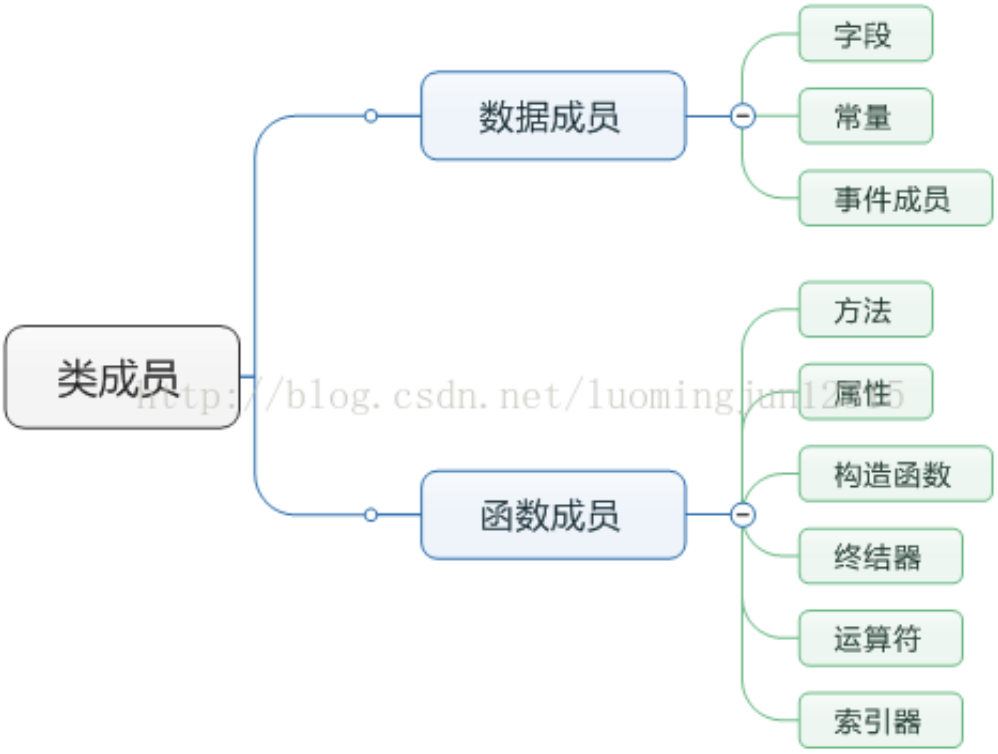
在看过C++之后，再看C#的面向对象感觉就不难了，只是有一些区别而已。那么现在我们来看看什么是类。类是面向对象语言和面向过程语言最大的区别。然而抽象就是面向对象的基本方法。对于抽象我们一点都不陌生，因为抽象是人类认识问题的基本手段之一。抽象是指对具体问题进行概括，抽出一类对象的公共性质并加以描述的过程。

一般一个问题的抽象应该包含两个方面：数据抽象和行为抽象。于是在面向对象语言，就引入了类的概念，所以说类就是数据抽象和行为抽象的集合。也就是说类就相当于，一个模具。我们可以通过这个模具制造出很多一样的产品。而这个产品就是对象。换而言之，对象就是类的具体化。由于类和对象的加入，使得我们的开发效率大大提高。下面我们来看看类的具体内容吧。

1、类定义

使用class关键字来声明类，其和C++不同的地方是在大括号之后不需要冒号

C#2、类成员



3、字段与属性

首先我们先区分一下C#数据成员中的字段、常量与事件成员。字段、常量是与类的相关变量。事件是类的成员，在发生某些行为时（如：改变类的字段或属性，或进行某种形式的用户交互操作），它可以让对象通知调用方。

那么现在我们在来看看字段与属性，属性的定义如下：

C#

```
[访问权限][类型][属性名称]
{
        //get和set前访问属性默认为public
        get
        {
                return [该属性对应的字段];
        }
        set
        {
                //作你想对该字段进行的操作
        }
}
```

下面我们来看看，实例，顺便了解一下get和set的原理

【C#】

```
class Test
{
        private int num;
        public int getNum()
        {
                return num;
        }
        public void setNum(int value)
        {
                num = value;
        }
}
```

不知道你发现了没有，在C#中，将getNum这个函数做成了get,setNum做成了set，将set得到的值用value保存。其实说白了，属性就是可以对某一字段进行读取和赋值操作的方法。但是调用的时候就不需要方法名称后的括号。就以上面的代码为例： Test.Num 。只要这样就可以调用他了。

4、方法

其实这个和就是成员函数，不过在C#中成员函数的范围更广。现在我们来看看

方法的定义

```
[访问属性][返回值类型] 方法名称([参数])
{
        //方法内部实现
}
```

举个例子：

```
using System;
namespace MyFirstProgram
{
        class Program
        {
                class Test
                {
                        private int a,b;
                        public int Add()
                        {
                                return a + b;
                        }
                }
                static void Main()
                {
                        Test p = new Test();
                        p.Add(); //对方法的调用
                }
        }
}
```

给方法传递参数

这个就和普通的函数传递参数是一样的。在C#中，特别注意，除非特别声明，否则引用类型均为引用传递，值类型均为值传递。

可选参数

其实这个就是C++中的默认值

ref 与 out 参数

值传递的类型是默认的，但是有些时候我们需要对他进行改变。这时就出现了 ref 与 out。

C#

```
public void Add(ref int b)
{
        b++;
}
```

调用时还需要添加ref关键字

```
p.Add(ref i);
```

其实这就和C++中的引用机制一样，不同的是，在C#中 ref 必须 要使用赋过值的变量。然而out为前缀

时，变量可以不初始化。

特别注意：ref 与 out 的 参数不能带有默认值：

C#

```
1
2
3
4
public void Add(ref int b = 2)    //错误，对其进行默认值，所以不能用ref或out
{
        b++;
}
```

无序传递

这个机制，在C++中没有。不过觉得貌似也没什么用

```
C#//函数
string setName(string FName,string LName){
        return FName + " " + LName;
}//正常调用
setName("John","Doe");
//无序调用
setName(LName: "Doe", FName: "John");
```

两种调用得到的结果是一致的。

5、构造函数

构造函数是在创建对象时，对对象数据初始化的方法。其方法名称与类名相同，且没有返回类型。如果没有写的话，编译器会自动创建一个无参构造函数，将所有的数据初始化为标准的默认值。

C#

```
class Test
{
        //只要创建了无参构造函数，后台将不会自动生成构造函数
        public Test()
        {
                //构造函数实现
        }
        //重写构造函数
        public Test(int Number)
        {
                //构造函数内部实现
        }
}
```

C#还有一个特性是可以给类编写无参数的构造函数，编写静态构造函数的原因就是，类内部有一些静态字段和属性，需要在第一次使用类之前初始化。注意：静态构造函数至多只运行一次。

6、匿名类型

由关键字var声明，其隐式类型化的变量。其实我觉得这个就相当于一个没有名称的类一样。这个感觉并不好用，所以还是慎用。声明如下：

如果一个对象包含一个人的姓名。

C#7、部分类

使用 partial 关键字可以将类、结构方法或借口放在多个文件中，通常一个类在一个文件中，但是在多人开发的时候就可以体现出partial的威力了。

8、弱引用

这个机制，是为了节约内存。因为当程序实例化一个类或结构时，只要有代码引用就会，形成强引用（相对弱引用而言）。如果这个类十分巨大，且不经常访问。那么就会浪费大量的内存，所以就引入了弱引用的概念。弱引用是使用WeakReference 类创建的。因为对象随时可能被回收所以在引用该对象时必须先确认它的存在。

C#

```
static void Main()
{
        WeakReference TestReference = new WeakReference(new Test());
        Test tmp;
        //IsAlive这个属性的目的就是TestReference的对象被回收了没有
        if (TestReference.IsAlive)
        {
                //如果类对象存在，那就引用该对象
                tmp = TestReference.Target as Test;
        }
```

}9、静态类

静态类的内部只用静态的方法和属性。静态类在功能上与 使用私有静态构造函数创建的类相同。注意：静态类不能创建实例。

反之我们可以使用static关键字检查是否创建了该类的实例，如果是编译器就会报错。

10、类（class）与结构（struct）的区别

①类可以继承，然而结构不能

②类储存在托管堆上，然而结构储存在栈上（所以结构存储效率优于类）

③类是引用类型，而结构是值类型

④原则结构的默认无参构造函数不可替换，但是如果将无参构造函数重写是对全体数据成员进行赋值的话，那么就可以重写

11、只读字段与常量的区别

只读属性有readonly声明。常量概念就是包含一个不可修改的变量。但是有时候需要一些变量其值不能改变，但运行之前是未知的。所以我们就引入了只读属性。只读属性比const灵活很多，readonly可以再构造函数中给其赋值，所以说每个实例都可以有不同的值。

12、typeof的使用

typeof可以获取对象类型，

C#