

静态类用于创建无需创建类的实例就能访问的数据和函数，不依赖特定对象的存在，静态类只能包含静态成员，不可以包含非静态成员，访问静态成员采用类名加点操作加静态成员

静态类是不能实例化的，我们采用类名加点操作符加变量名直接使用它的属性与方法

静态方法被调用的时候是通过**类名**访问的，比如Users类中有一个静态方法Add()，调用的时候可以直接这个写 Users.Add();

类可以声明为 static 的，以指示它仅包含静态成员。函数成员和变量都必须有static修饰

实例方法的存在必须要有对象实例的存在，如果对象实例不存在，则实例方法也就没有调用它的主人。静态方法的存在前提是类的存在，所以无需声明和New对象

**第一：**C#静态成员：（1）静态成员属于类所有，非静态成员属于类的实例所有。

                                     （2）每创建一个类的实例，都会在内存中为非静态成员新分配一块存储。

           总结：静态成员属于类所有，为各个类的实例所公用，无论类创建了多少实例，类的静态成员在内存中只占同一块区域。

**第二：**C#静态方法:     (1)C#静态方法属于类所有，类实例化前即可使用。

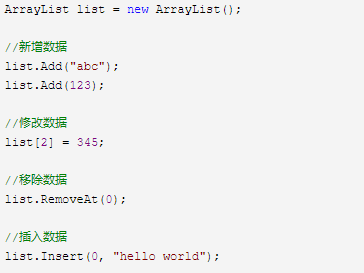
                                    （2）非静态方法可以访问类中的任何成员，静态方法只能访问类中的静态成员。

扩展或修改继承的方法、属性、索引器或事件的抽象实现或虚实现，必须使用 override 修饰符。

ArrayList：

ArrayList是.Net Framework提供的用于数据存储和检索的专用类，它是命名空间System.Collections下的一部分。它的大小是按照其中存储的数据来动态扩充与收缩的。所以，我们在声明ArrayList对象时并不需要指定它的长度。

ArrayList继承了IList接口，所以它可以很方便的进行数据的添加，插入和移除.比如：



从上面示例看，ArrayList好像是解决了数组中所有的缺点，那么它应该就是完美的了，为什么在C#2.0后又会出现List呢？

还是从上面的示例看，在list中，我们不仅插入了字符串"abc"，而且又插入了数字123。这样在ArrayList中插入不同类型的数据是允许的。因为ArrayList会把所有插入其中的数据都当作为object类型来处理。这样，在我们使用ArrayList中的数据来处理问题的时候，很可能会报类型不匹配的错误，也就是说ArrayList不是类型安全的。既使我们保证在插入数据的时候都很小心，都有插入了同一类型的数据，但在使用的时候，我们也需要将它们转化为对应的原类型来处理。这就存在了装箱与拆箱的操作，会带来很大的性能损耗。

而List类是ArrayList类的泛型等效类。它的大部分用法都与ArrayList相似，因为List类也继承了IList接口。最关键的区别在于，在声明List集合时，我们同时需要为其声明List集合内数据的对象类型。

这是创建了一个只能装string类型的泛型集合。

也可以理解为一个高级的数组，List<T>比普通的数组拥有更多的属性和方法。能完成更复杂的业务需求。

比如普通的数组必须定义长度且长度是固定的，而List<T>是可以随意改变长度的。

List<T> 是泛型列表，T可以是任意类型，用来存放T类型对象的一个不限长度列表。

List<string> listdata=new List<string> ()

意思是声明一个名字为listdata的字符串泛型列表并初始化。

可以用下标 Listdata[i] 访问存到列表中的某个元素。Listdata.add(strValue)可以将strValue这个字符串对象添加到Listdata列表中。

dataset是数据集，他内部可以包含多个datatable对象和数据集本身的信息。就像学校。

datatable是数据表，他内部包含多个datarow对象 ，就像是班级，

datarow 是数据行对象，他内部有包含多个数据列。 就像是某个班级里座位的行

C#中的catch和finally：

1、将预见可能引发异常的代码包含在try语句块中。

2、如果发生了异常，则转入catch的执行。

catch有几种写法：

catch 这将捕获任何发生的异常。

catch(Exception e) 这将捕获任何发生的异常。另外，还提供e参数，你可以在处理异常时使用e参数来获得有关异常的信息。

catch(Exception的派生类 e) 这将捕获派生类定义的异常，例如安卓中文网，我想捕获一个无效操作的异常，可以如下写：

catch(InvalidOperationException e) { .... } 这样，如果try语句块中抛出的异常是InvalidOperationException，将转入该处执行，其他异常不处理。

catch可以有多个，也可以没有，每个catch可以处理一个特定的异常。.net按照你catch的顺序查找异常处理块，如果找到，则进行处理，如果找不到，则向上一层次抛出。如果没有上一层次，则向用户抛出，此时，如果你在调试，程序将中断运行，如果是部署的程序，将会中止。 如果没有catch块，异常总是向上层（如果有）抛出，或者中断程序运行。