花纸de小窝

C#面向对象。

2020-5-15 23:10 | 85 | 0 | C#

面向对象介绍。

解释

面向对象编程——Object Oriented Programming，简称OOP，是一种程序设计思想。OOP把对象作为程序的基本单元，一个对象包含了数据和操作数据的函数。

面向过程的程序设计把计算机程序视为一系列的命令集合，即一组函数的顺序执行。为了简化程序设计，面向过程把函数继续切分为子函数，即把大块函数通过切割成小块函数来降低系统的复杂度。

而面向对象的程序设计把计算机程序视为一组对象的集合，而每个对象都可以接收其他对象发过来的消息，并处理这些消息，计算机程序的执行就是一系列消息在各个对象之间传递。

类

类就是个模子，确定对象应该具有的属性（特征）和行为（方法）。

语法：
public class 类名
{
字段;
属性;
方法;
}

写好了一个类之后，我们需要创建这个类的对象，
那么，我们管创建这个类的对象过程称之为类的实例化。
使用关键字 new.

this:表示当前这个类的对象。
类是不占内存的，而对象是占内存的。

属性

属性的作用就是保护字段、对字段的赋值和取值进行限定。
属性的本质就是两个方法，一个叫get()一个叫set()。
既有get()也有set()我们诚之为可读可写属性。
只有get()没有set()我们称之为只读属性
没有get()只有set()我们称之为只写属性
在set方法中判断Value的值，在get方法中判断字段的值

Field 字段
Method 方法
Property 属性

访问修饰符

public：公开的公共的，在哪都能访问。
private：私有的，只能在当前类的内部进行访问，出了这个类就访问不到了。

静态与非静态的区别

1、在非静态类中，既可以有实例成员，也可以有静态成员。
2、在调用实例成员的时候，需要使用对象名.实例成员;
在调用静态成员的时候，需要使用类名.静态成员名;

总结：静态成员必须使用类名去调用，而实例成员使用对象名调用。
静态函数中，只能访问静态成员，不允许访问实例成员。
实例函数中，既可以使用静态成员，也可以使用实例成员。
静态类中只允许有静态成员，不允许出现实例成员。

使用：
1、如果你想要你的类当做一个”工具类”去使用，这个时候可以考虑将类写成静态的。
2、静态类在整个项目中资源共享。
只有在程序全部结束之后，静态类才会释放资源。

堆 栈 静态存储区域

释放资源。GC Garbage Collection垃圾回收器

构造函数

作用：帮助我们初始化对象(给对象的每个属性依次的赋值)
构造函数是一个特殊的方法：
1、构造函数没有返回值，连void也不能写。
2、构造函数的名称必须跟类名一样。

创建对象的时候会执行构造函数
构造函数是可以有重载的。

类当中会有一个默认的无参数的构造函数，当你写一个新的构造函数之后，不管是有参数的还是无参数的，那个默认的无参数的构造函数都被取代了。

new关键字

Person zsPerson=new Person();

new帮助我们做了3件事儿：

1、在内存中开辟一块空间

2、在开辟的空间中创建对象

3、调用对象的构造函数进行初始化对象

this关键字

1、代表当前类的对象

2、在类当中显示的调用本类的构造函数 :this

面向对象继承

命名空间

可以认为类是属于命名空间的。

如果在当前项目中没有这个类的命名空间，需要我们手动的导入这个类所在的命名空间。

1、用鼠标去点

2、alt+shift+F10

3、记住命名空间，手动的去引用

4、在一个项目中引用另一个项目的类

4.1、添加引用

4.2、引用命名空间

值类型和引用类型

区别：

1、值类型和引用类型在内存上存储的地方不一样。

2、在传递值类型和传递引用类型的时候，传递的方式不一样。

值类型我们称之为值传递，引用类型我们称之为引用传递。

我们学的值类型和引用类型：

值类型：int、double、bool、char、decimal、struct、enum(枚举)

引用类型：string、自定义类、数组

存储：

值类型的值是存储在内存的栈当中。

引用类型的值是存储在内存的堆中。

值类型和引用类型在内存上存储的地方不一样。

字符串

1、字符串的不可变性

当你给一个字符串重新赋值之后，老值并没有销毁，而是重新开辟一块空间存储新值。

当程序结束后，GC扫描整个内存，如果发现有的空间没有被指向，则立即把它销毁。

2、我们可以讲字符串看做是char类型的一个只读数组。

ToCharArray();将字符串转换为char数组

new string(char[] chs):能够将char数组转换为字符串

字符串提供的各种方法

1、Length：获得当前字符串中字符的个数

2、ToUpper():将字符转换成大写形式

3、ToLower():将字符串转换成小写形式

4、Equals(lessonTwo,StringComparison.OrdinalIgnoreCase):比较两个字符串，可以忽略大小写

5、Split()：分割字符串，返回字符串类型的数组。

6、Substring()：截取字符串。在截取的时候包含要截取的那个位置。

7、IndexOf():判断某个字符串在字符串中第一次出现的位置，如果没有返回-1、

8、LastIndexOf()：判断某个字符串在字符串中最后一次出现的位置，如果没有同样返回-1

9、StartsWith():判断以....开始

10、EndsWith():判断以...结束

11、Replace():将字符串中某个字符串替换成一个新的字符串

12、Contains():判断某个字符串是否包含指定的字符串

13、Trim():去掉字符串中前后的空格

14、TrimEnd()：去掉字符串中结尾的空格

15、TrimStart()：去掉字符串中前面的空格

16、string.IsNullOrEmpty():判断一个字符串是否为空或者为null

17、string.Join()：将数组按照指定的字符串连接，返回一个字符串。

继承

我们可能会在一些类中，写一些重复的成员，我们可以将这些重复的成员，单独的封装到一个类中，作为这些类的父类。
Student、Teacher、Driver 子类 派生类
Person 父类 基类
子类继承了父类，那么子类从父类那里继承过来了什么？
首先，子类继承了父类的属性和方法，但是子类并没有继承父类的私有字段。
问题：子类有没有继承父类的构造函数？
答：子类并没有继承父类的构造函数，但是。子类会默认的调用父类无参数的构造函数，
创建父类对象，让子类可以使用父类中的成员。
所以，如果在父类中重新写了一个有参数的构造函数之后，那个无参数的就被干掉了，
子类就调用不到了，所以子类会报错。
解决办法：
1、在父类中重新写一个无参数的构造函数。
2、在子类中显示的调用父类的构造函数，使用关键字:base()

继承的特性

1、继承的单根性：一个子类只能有一个父类。
2、继承的传递性

Object类是所有类的基类。所有类都直接或者间接继承Object。

new关键字：
1、创建对象
2、隐藏从父类那里继承过来的同名成员。
隐藏的后果就是子类调用不到父类的成员。

里氏转换

1、子类可以赋值给父类，如果有一个地方需要一个父类作为参数，我们可以给一个子类代替
2、如果父类中装的是子类对象，那么可以讲这个父类强转为子类对象。

子类对象可以调用父类中的成员，但是父类对象永远都只能调用自己的成员。

is：表示类型转换，如果能够转换成功，则返回一个true，否则返回一个false

```
        if (p is Student)
        {
            Student ss = (Student)p;
            ss.StudentSayHello();
        }
        else
        {
            Console.WriteLine("转换失败");
        }
```

as：表示类型转换，如果能够转换则返回对应的对象，否则返回一个null

我们将一个对象输出到控制台 默认情况下 打印的就是这个对象所在的类的命名空间。

protected

受保护的：可以在当前类的内部以及该类的子类中访问。

ArrayList集合的长度问题

每次集合中实际包含的元素个数(count)超过了可以包含的元素的个数(capcity)的时候，
集合就会向内存中申请多开辟一倍的空间，来保证集合的长度一直够用。

Console.WriteLine(list.Count); count 表示这个集合中实际包含的元素的个数
Console.WriteLine(list.Capacity); capcity 表示这个集合中可以包含的元素的个数

ArrayList的方法

```
        添加单个元素
        list.Add(true);
        添加集合元素
        list.AddRange(new int[] { 1, 2, 3, 4, 5, 6, 7, 8, 9 });
        list.AddRange(list);
        list.Clear(); 清空所有元素
        list.Remove(true); 删除单个元素 写谁就删谁
        list.RemoveAt(0); 根据下标去删除元素
        list.RemoveRange(0, 3); 根据下标去移除一定范围的元素
        list.Sort();//升序排列，起码可以比较的时候
        list.Reverse(); 反转
        list.Insert(1, "插入的"); 在指定的位置插入一个元素
```

```
13          list.InsertRange(0, new string[] { "张三", "李四" }); 在指定的位置插入一个集合
14          bool b = list.Contains(1); 判断是否包含某个指定的元素
```


Hashtable 键值对集合

在键值对集合当中，我们是根据键去找值的。

键值对对象[键]=值;

键值对集合当中，键必须是唯一的，而值是可以重复的

var:根据值能够推断出来类型 var n = 1 那么n的数据类型就是int

path类

```
1           string str = @"F:\unrealEngine\unrealEngineWordSpace\Demo1ForAnniversary\asd\1.txt"
2           //获得文件名
3           Console.WriteLine(Path.GetFileName(str));
4           //获得文件名但是不包含扩展名
5           Console.WriteLine(Path.GetFileNameWithoutExtension(str));
6           //获得文件的扩展名
7           Console.WriteLine(Path.GetExtension(str));
8           //获得文件所在的文件夹的名称
9           Console.WriteLine(Path.GetDirectoryName(str));
10          //获得文件所在的全路径
11          Console.WriteLine(Path.GetFullPath(str));
```

```
12          //连接两个字符串作为路径
13          Console.WriteLine(Path.Combine(@"c:\a\" , "b.txt"));
```


file类方法

```
1       //读取指定文件的内容
2       byte[] buffer = File.ReadAllBytes(@"C:\Users\SpringRain\Desktop\抽象类特点.txt");
3       //将字节数组中的每一个元素都要按照我们指定的编码格式解码成字符串
4       //UTF-8  GB2312 GBK ASCII  Unicode
5       string s = Encoding.Default.GetString(buffer);
```

```
1       //没有这个文件的话 会给你创建一个 有的话 会给你覆盖掉
2       string str = "今天天气好晴朗处处好风光";
3       //需要将字符串转换成字节数组
4       byte[] buffer = Encoding.Default.GetBytes(str);
5       //然后把指定字符串写入指定文件内
6       File.WriteAllBytes(@"C:\Users\SpringRain\Desktop\new.txt", buffer);
```

```
1  //按指定格式读取文件   按行读    返回字符数组
```

```
string[] contents = File.ReadAllLines(@"C:\Users\a.txt", Encoding.Default);
//按指定格式读取文件，返回成字符串。
string str = File.ReadAllText("抽象类特点.txt", Encoding.Default);
//按指定格式读取文件，返回字节数组
byte[] buffer = File.ReadAllBytes(@"C:\Users\SpringRain\Desktop\抽象类特点.txt");
//将指定字符数组写入指定文件内，按行写入
File.WriteAllLines(@"C:\Users\SpringRain\Desktop\new.txt", new string[] { "aoe", "ewu" });
//将指定字符串写入指定文件内
File.WriteAllText(@"C:\Users\SpringRain\Desktop\new.txt", "张三李四王五赵六");
//在指定文件内容后面追加字符串。
File.AppendAllText(@"C:\Users\SpringRain\Desktop\new.txt", "看我有木有把你覆盖掉");
```

花纸de小

```
            //创建一个文件
            File.Create(@"C:\Users\SpringRain\Desktop\new.txt");
            Console.WriteLine("创建成功");
            Console.ReadKey();

            //删除一个文件
            File.Delete(@"C:\Users\SpringRain\Desktop\new.txt");
            Console.WriteLine("删除成功");
            Console.ReadKey();
            //1024byte=1kb
            //1024kb=1M
            //1024M=1G
            //1024G=1T
```

```
14              //1024T=1PT
15                                                                          花纸de小
16              //复制一个文件
17              File.Copy(@"C:\Users\SpringRain\Desktop\code.txt", @"C:\Users\SpringRain\Desktop\new
18              Console.WriteLine("复制成功");
19              Console.ReadKey();
20
21
22              //剪切
23              File.Move(@"C:\Users\SpringRain\Desktop\code.txt", @"C:\Users\SpringRain\Desktop\new
24              Console.WriteLine("剪切成功");
25              Console.ReadKey();
```

面向对象多态

绝对路径和相对路径

绝对路径：通过给定的这个路径直接能在我的电脑中找到这个文件。

相对路径：文件相对于应用程序的路径。 项目名称\bin\Debug 下

结论：

我们在开发中应该去尽量的使用相对路径。

装箱和拆箱

装箱：就是将值类型转换为引用类型。

拆箱：将引用类型转换为值类型。

看两种类型是否发生了装箱或者拆箱，要看，这两种类型是否存在继承关系。

FileStream

```
1               //使用FileStream来读取数据                路径          文件模式：打开或创建     处理数据：读取
2               FileStream fsRead = new FileStream(@"D:\a.txt", FileMode.OpenOrCreate, FileAccess.R
3               byte[] buffer = new byte[1024 * 1024 * 5];
4               //3.8M  5M
5               //返回本次实际读取到的有效字节数  r
6               //                  字节数组  从哪儿读    最多读的长度
7               int r = fsRead.Read(buffer, 0, buffer.Length);
8               //将字节数组中每一个元素按照指定的编码格式解码成字符串
9               //                                      解码这个    从0开始   解码R个
10              string s = Encoding.Default.GetString(buffer, 0, r);
11              //关闭流
12              fsRead.Close();
13              //释放流所占用的资源
14              fsRead.Dispose();
15              Console.WriteLine(s);
16              Console.WriteLine(r);
17              Console.ReadKey();
```

将创建文件流对象的过程写在using当中，会自动的帮助我们释放流所占用的资源。

```
//使用FileStream来写入数据
            using (FileStream fsWrite = new FileStream(@"D:\a.txt", FileMode.OpenOrCreate, FileA
            {
                string str = "看我又把你覆盖掉";
                byte[] buffer = Encoding.Default.GetBytes(str);
                fsWrite.Write(buffer, 0, buffer.Length);
            }
            Console.WriteLine("写入OK");
            Console.ReadKey();
```



```
//复制代码
public static void CopyFile(string soucre, string target)
        {
            //1、我们创建一个负责读取的流
            using (FileStream fsRead = new FileStream(soucre, FileMode.Open, FileAccess.Read))
            {
                //2、创建一个负责写入的流
                using (FileStream fsWrite = new FileStream(target, FileMode.OpenOrCreate, FileAc
                {
                    byte[] buffer = new byte[1024 * 1024 * 5];
                    //因为文件可能会比较大，所以我们在读取的时候 应该通过一个循环去读取
                    while (true)
                    {
                        //返回本次读取实际读取到的字节数
```

```
15                      int r = fsRead.Read(buffer, 0, buffer.Length);
16                      //如果返回一个0，也就意味什么都没有读取到，读取完了
17                      if (r == 0)
18                      {
19                          break;
20                      }
21                      fsWrite.Write(buffer, 0, r);
22                  }
23
24
25              }
26          }
27      }
```


StreamReader和StreamWriter

操作文本类比较简单。不用创建数组。

```
1           //使用StreamReader来读取一个文本文件
2           using (StreamReader sr = new StreamReader(@"C:\Users\SpringRain\Desktop\抽象类特点.tx
3           {
4               while (!sr.EndOfStream)
5               {
6                   Console.WriteLine(sr.ReadLine());
7               }
```

```
8               }
```



```
1               //使用StreamWriter来写入一个文本文件
2               using (StreamWriter sw = new StreamWriter(@"C:\Users\SpringRain\Desktop\newnew.txt",
3               {
4                   sw.Write("看我有木有把你覆盖掉");
5               }
6               Console.WriteLine("OK");
7               Console.ReadKey();
```

多态

概念：让一个对象能够表现出多种的状态（类型）

实现多态的有3种手段

虚方法(virtual)

将父类的方法标记为虚方法，使用关键字 virtual，这个函数可以被子类重新(override)写一个遍。

```
1  //父类
2          public virtual void SayHello()
3          {
```

```
4               Console.WriteLine("我是父类");
5           }
6   //子类
7           public override void SayHello()
8           {
9               Console.WriteLine("我是中国人，我叫{0}", this.Name);
10          }
```


抽象类

当父类中的方法不知道如何去实现的时候，可以考虑将父类写成抽象类，将方法写成抽象方法。

```
1       //关健词 abstract
2       public abstract class Animal
3       {
4           //抽象方法是没有方法体的
5           public abstract void Bark();
6       }
7       public class Dog : Animal {
8           //进行重写。
9           public override void Bark()
10          {
11              Console.WriteLine("狗在汪汪叫");
12          }
13      }
```

抽象类特点：

1.抽象成员必须标记为abstract,并且不能有任何实现。
2.抽象成员必须在抽象类中。
3.抽象类不能被实例化

4.子类继承抽象类后，必须把父类中的所有抽象成员都重写。

（除非子类也是一个抽象类，则可以不重写）
5.抽象成员的访问修饰符不能是private
6.在抽象类中可以包含实例成员。
并且抽象类的实例成员可以不被子类实现

7.抽象类是有构造函数的。虽然不能被实例化。

8.如果父类的抽象方法中有参数，那么。继承这个抽象父类的子类在重写父类的方法的时候必须传入对应的参数。

如果抽象父类的抽象方法中有返回值，那么子类在重写这个抽象方法的时候 也必须要传入返回值。

======
如果父类中的方法有默认的实现，并且父类需要被实例化，这时可以考虑将父类定义成一个普通类，用虚方法来实现多态。

如果父类中的方法没有默认实现，父类也不需要被实例化，则可以将该类定义为抽象类。

传参 有好几种方式：
第一种是在方法括号内进行传参
第二种是在类里面进行封装参数，然后给参数赋值。

接口

接口是一种规范。一种能力
只要一个类继承了一个接口，这个类就必须实现这个接口中所有的成员

为了多态。
接口不能被实例化。
也就是说，接口不能new(不能创建对象)

接口中的成员不能加“访问修饰符”，接口中的成员访问修饰符为public,不能修改。

（默认为public）
接口中的成员不能有任何实现（“光说不做”，只是定义了一组未实现的成员）。

接口中只能有方法、属性、索引器、事件，不能有“字段”和构造函数。

接口与接口之间可以继承，并且可以多继承。

接口并不能去继承一个类，而类可以继承接口 （接口只能继承于接口，而类既可以继承接口，也可以继承类）

实现接口的子类必须实现该接口的全部成员。

一个类可以同时继承一个类并实现多个接口，如果一个子类同时继承了父类A，并实现了接口IA,那么语法上A必须写在IA的前面。

class MyClass:A,IA{}，因为类是单继承的。

显示实现接口的目的：解决方法的重名问题
什么时候显示的去实现接口：
当继承的借口中的方法和参数一摸一样的时候，要是用显示的实现接口

当一个抽象类实现接口的时候，需要子类去实现接口。

访问修饰符

public :公开的公共的
private：私有的，只能在当前类的内部访问
protected：受保护的，只能在当前类的内部以及该类的子类中访问。
internal：只能在当前项目中访问。在同一个项目中，internal和public的权限是一样。
protected internal：protected+internal

1)、能够修饰类的访问修饰符只有两个：public、internal。
2)、可访问性不一致。
子类的访问权限不能高于父类的访问权限，会暴漏父类的成员

值类型和引用类型

值类型:int double char decimal bool enum struct 存储在栈内
引用类型：string 数组 自定义类 集合 object 接口 存储在堆中

值类型复制的时候传递是这个值本身
引用类型在复制的时候，传递的是对这个对象的引用。
ref 把值传递转换成引用传递

序列化

序列化：就是将对象转换为二进制
反序列化：就是将二进制转换为对象
作用：传输数据。

序列化步骤
先将这个类标记为可以被序列化的。 标记：[Serializable]

```
        //要将p这个对象 传输给对方电脑
            Person p = new Person();
            p.Name = "张三";
            p.Age = 19;
            p.Gender = '男';
            using (FileStream fsWrite = new FileStream(@"C:\Users\SpringRain\Desktop\111.txt",
            {
                //开始序列化对象
                BinaryFormatter bf = new BinaryFormatter();
                                //流 要序列化的对象。
                bf.Serialize(fsWrite, p); 自动写入流了。
            }
            Console.WriteLine("序列化成功");
            Console.ReadKey();
```

反序列化

```
        //接收对方发送过来的二进制 反序列化成对象
        Person p;
        using (FileStream fsRead = new FileStream(@"C:\Users\SpringRain\Desktop\111.txt", FileM
        {
            BinaryFormatter bf = new BinaryFormatter();
            //反序列化
            p = (Person)bf.Deserialize(fsRead);
        }
        Console.WriteLine(p.Name);
        Console.WriteLine(p.Age);
        Console.WriteLine(p.Gender);
        Console.ReadKey();
```


部分类

部分类：partial 表示一个类的一部分。

密封类

密封类：sealed

不能够被其他类继承，但是可以继承于其他类。

接口

[public] interface I..able
{
成员;
}

接口中的成员不允许添加访问修饰符，默认就是public

显示实现接口

显示实现接口就是为了解决方法的重名问题

```
1       public class Bird : IFlyable
2       {
3           public void Fly()
4           {
5               Console.WriteLine("鸟飞会");
6           }
7           /// <summary>
8           /// 显示实现接口
9           /// </summary>
10           void IFlyable.Fly()
11          {
12              Console.WriteLine("我是接口的飞");
13          }
14
```

```
    }

    public interface IFlyable
    {
        void Fly();
    }
```


参考资料

C#指南。

dotNet文档。

C# 面向对象

上一篇

C#飞行棋小练习（我是猪。做不好）

下一篇

Key&Peele黑人兄弟第四季中英双字幕——缓慢更新中。

Theme **Argon**