里氏替换原则，派生类（子类）对象能够替换基类（超类）对象被使用

2014-03-20 07:40 950人阅读 评论(0) 收藏[1] 举报

当一个父类的变量指向一个子类对象的时候只能通过这个父类变量调用父类成员，子类独有的成员无法调用。？？

首先，里氏替换原则。

这是理解多态所必须掌握的内容。对于里氏替换原则维基百科给出的定义如下：

里氏替换原则的内容可以描述为：“派生类（子类）对象能够替换其基类（超类）对象被使用。”以上内容并非利斯科夫的原文，而是译自罗伯特·马丁（Robert Martin）对原文的解读。其原文为：

为什么子类可以替换父类的位置，而程序的功能不受影响呢？

当满足继承的时候，父类肯定存在非私有成员，子类肯定是得到了父类的这些非私有成员（假设，父类的的成员全部是私有的，那么子类没办法从父类继承任何成员，也就不存在继承的概念了）。既然子类继承了父类的这些非私有成员，那么父类对象也就可以在子类对象中调用这些非私有成员。所以，子类对象可以替换父类对象的位置。

来看下面的一段代码：

```
class Program
   {
       static  Main(string[] args)
       {
           Person p =  Person();

           Person p1 =  Student();

           Console.ReadKey();
       }
   }

   class Person
   {
       //父类的私有成员
       private int nAge;


       public Person()
       {
           Console.WriteLine(我是Person构造函数，我是一个人！public  Say()
       {
           Console.WriteLine(我是一个人！class Student : Person
   {
       public Student()
       {
           Console.WriteLine(我是Student构造函数，我是一个学生！public  SayStude()
       {
           Console.WriteLine(我是一个学生！class SeniorStudent : Student
   {
       public SeniorStudent()
       {
           Console.WriteLine(我是SeniorStudent构造函数，我是一个高中生！public   SaySenior()
       {
```

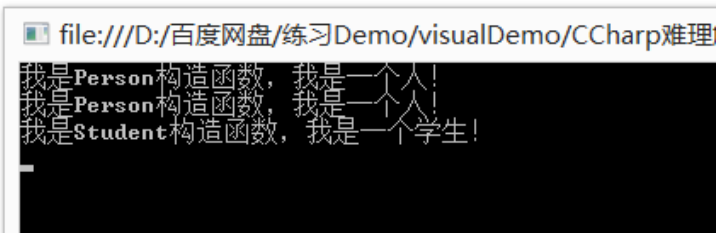
```
        Console.WriteLine(我是一个高中生!
```

我们运行打印出的结果是：

```
0 个引用
static void Main(string[] args)
{
    Person p = new Person();

    Person p1 = new Student();

    Console.ReadKey();
}
```



二，虚方法

使用virtual关键字修饰的方法，叫做虚方法（一般都是在父类中）。

看下面的一段代码：

```
class Person
    {
        private  nAge;
        public Person()
        {
            Console.WriteLine(我是Person构造函数，我是一个人！这里定义了一个虚方法
        public virtual  Say()
        {
            Console.WriteLine(我是一个人！class Student : Person
    {
        子类使用override关键字改写了父类的虚方法
        public override  Say()
        {
            Console.WriteLine(我是一个学生！public Student()
        {
            Console.WriteLine(我是Student构造函数，我是一个学生！public  SayStude()
        {
            Console.WriteLine(我是一个学生!
```

紧接着在main（）函数中添加如下的代码：

```
static  Main(string[] args)
        {
            Person p =  Person();
            p.Say();

            Person p1 =  Student();
            p1.Say();

            Student s =  Student();
            s.Say();
            Console.ReadKey();
        }
```

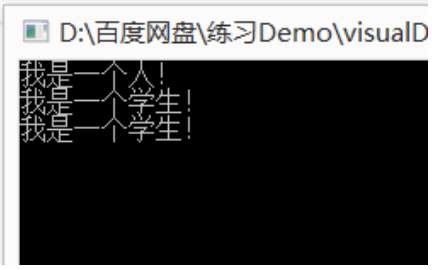
打印结果如下：

```
{
    Person p = new Person();
    p.Say();

    Person p1 = new Student();
    p1.Say();

    Student s = new Student();
    s.Say();
    Console.ReadKey();
}
```

这里我为了只打印方法里的内容，注释掉了构造函数



我们很明显的可以发现，第二个表达式满足里氏替换原则，p1.Say()执行的应该是父类的Say()方法，但是这里却执行了子类的Say()方法。

这就是子类使用override关键字的Say()方法覆盖了父类的用Virtual关键字修饰的Say()方法。

使用as关键字和第一种强制转换的区别就是，第一种如果转换失败会抛异常，第二种转换失败则返回一个null值。

Links

1. javascript:void(0);