继承

一、继承介绍

1. 子类可以继承父类的属性和方法，但是不能继承私有类型
2. 子类没有继承父类的构造函数，但在创建子类对象时(无论使用子类的有参还是无参构造)，会首先调用父类的无参构造，然后再调用子类的构造函数。

1. 继承的实质

子类使用父类的成员时，是先在子类中new了一个父类的对象，然后用这个隐藏对象去调用父类的方法（该对象是实际存在的，占内存），因此new子类对象时必定会调用父类的无参构造

```
    public class Human{
        public Human(){
            System.Console.WriteLine("我是Human的无参构造");
        }
    }
    public class Student : Human{
        public Student(){
            System.Console.WriteLine("我是Student的无参构造");
        }
    }
    static void Main(string[] args){
        Student stu1 = new Student();
    }
    //运行结果为:
    //我是Human的无参构造
    //我是Student的有参构造
```

2. 若父类中没有无参构造(排除默认构造的情况)，子类要调用时就会报错，解决办法:

1. 在父类中添加无参构造
2. 在子类中调用父类的有参构造，关键字: base()

```
    public class Human{
        public Human(string name,int age,char gender){
            this.Name = name;
            this.Age = age;
            this.Gender = gender;
        }
        public void CHLSS(){
            System.Console.WriteLine("我叫{0},今年{1}岁,是一名{2}生,我会吃喝拉撒
睡。",this.Name,this.Age,this.Gender);
        }
    }
    public class Student : Human{
```

```
        //id是Student自己的字段，其他三个是Human的字段
        public Student(string name,int age,char gender,int id) :
base(name,age,gender){
            this.Id = id;
        }
    }

    static void Main(string[] args){
        Student stu1 = new Student("小何",28,'男',2201794);
    }
    //运行结果
    //我叫小何,今年18岁,是一名男生,我会吃喝拉撒睡。
```

3. protected关键字

权限介于public和private之间，本类可以调用，而且其子类也可以调用

```
    public class Human{
        protected int test = 2;
    }

    public class Student : Human{
        public void stuTest(){
            //base. 表示调用父类中的成员，类似this. 调用本类成员
            System.Console.WriteLine(base.test);
        }
    }

    Student stu1 = new Student();
    stu1.stuTest();
    //结果输出 2
```

4. 继承的特性

1. 继承的单根性：一个类只能有一个父类
2. 继承的传递性：子类继承父类、父类继承爷类、爷类继承祖爷类……。且子类可以向上使用父类、爷类、祖爷类…的公用属性和方法
3. 父类对象不能向下调用子类对象的成员
4. 所有类都直接或间接继承object类(当定义一个类时没有指定继承，则默认继承object类)
5. 当父类和子类出现同名的方法时，调用该方法时会采用就近原则(子类调子类的，父类调父类的)。此时可在子类中使用**new**关键字，来隐藏父类中继承来的成员。

```
    public class Human{
        public void LikeDrive(){
            System.Console.WriteLine("我是父类中的LikeDrive方法");
```

```
        }
    }
    public class Driver : Human{
        public new void LikeDrive(){     //插入一个关键字new后可以避免警告
            System.Console.WriteLine("我是子类中的LikeDrive方法");
        }
    }
```

二、里氏转换*

1.里氏转换含义

1. 子类可以赋值给父类
   作用：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

```
Teacher tea1 = new Teacher();
Human hum1 = tea1;

//或者更简便:
Human hum2 = new Teacher();
```

2. 如果父类中装的是子类对象，那么可以将这个父类强转为子类对象
   注：转换前必须先将子类赋值给父类

```
//将父类强转成子类
Human hum1 = new Teacher();          //这句不能是new Human()
Teacher newTea = (Teacher)hum1;      //将hum1强转为Teacher类
newTea.SayHello();  //正常使用子类的方法
```

2.里氏转换注意事项

使用里氏转换时，容易搞错类从而出错，因此使用is和as做判断
注：转换前必须先将子类赋值给父类

1. is (如果能转换则返回true，否则返回false)

```
Human hum1 = new Student();
if(hum1 is Student){         //is : 判断前者是否能转换成后者
    Student stu1 = (Student)hum1;
    stu1.Study();
}else{
    System.Console.WriteLine("转换失败");
}
```

2. as (如果能转则返回对应的对象，否转则返回null)

```
    Human hum2 = new Teacher();
    Teacher tea1 = hum2 as Teacher;     //as : 能转换则直接转换，不能转换则返回null
    tea1.SayHello();
```

3.里氏转换练习

需求：创建多个子类对象装入父类数组中，然后随机调用所有子类自己的SayHello方法
注：对象装在human类数组中时，表现为human类，所以不能直接调用子类的方法

```
    Human[] humArray = new Human[10];        //创建一个Human型数组
    Random ran = new Random();               //创建一个随机数
    for (int i = 0; i < humArray.Length; i++)
    {
        int randomNum = ran.Next(1,5);       //产生一个1~4的随机数
        switch(randomNum){
            case 1 :
                humArray[i] = new Student();
                break;
            case 2:
                humArray[i] = new Teacher();
                break;
            case 3:
                humArray[i] = new Driver();
                break;
            case 4:
                humArray[i] = new Human();
                break;
        }
    }

    for (int i = 0; i < humArray.Length; i++)
    {
        //humArray[i].HumSayHello();       //因为对象全装在human类数组中，表现为human
类，所以不能直接调用子类的方法

        if(humArray[i] is Student){      //判断是否能强转
            //强转为Student类再调用Student的方法
            ((Student)humArray[i]).StuSayHello();
        }else if(humArray[i] is Teacher){

            ((Teacher)humArray[i]).TeaSayHello();
        }else if(humArray[i] is Driver){

            ((Driver)humArray[i]).DriSayHello();
        }else{
            humArray[i].HumSayHello();
        }
    }
```